AFRL-OSR-VA-TR-2013-0545

COMBINATORIAL MOTION PLANNING ALGORITHMS FOR A HETEROGENEOUS COLLECTION OF UNMANNED VEHICLES

SWAROOP DARBHA

TEXAS ENGINEERING EXPERIMENT STATION

10/15/2013
Final Report

AIR FORCE RESEARCH LABORATORY
AF OFFICE OF SCIENTIFIC RESEARCH (AFOSR)/RSL
ARLINGTON, VIRGINIA 22203
AIR FORCE MATERIEL COMMAND

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
13-10-2013	Final Report	15-07-2010 to 14-07-2013

4. TITLE AND SUBTITLE

Combinatorial Motion Planning Algorithms for a Heterogeneous Collection of Vehicles

5a. CONTRACT NUMBER

FA-9550-10-1-0392

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Swaroop Darbha, Sivakumar Rathinam and K. R. Rajagopal

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Texas Engineering Experimentation Station
College Station, TX-77843

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Distribution approved for public release

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The project dealt with two classes of core decision-making algorithms related to operator-UV collaboration; the first class involves the routing of UVs through the set of targets nominated by the operator and the second class of problems involves decision-making algorithms for UVs to accommodate uncertainty. We have developed approximation, lower bounding and exact algorithms to address the two classes of problems. We have also implemented these algorithms in simulations to corroborate the performance of these algorithms. In the ensuing discussion, we will summarize our work for the project, and our main results.

15. SUBJECT TERMS

Motion planning, routing, Unmanned Vehicles, Motion constraints, Dubins vehicles, approximation algorithms, transformation me

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**19b. TELEPHONE NUMBER** *(include area code)*

Final Report

October 13, 2013

COMBINATORIAL MOTION PLANNING ALGORITHMS FOR A HETEROGENEOUS COLLECTION OF UNMANNED VEHICLES (UVs)

FA 09550-10-1-0392

PI: Dr. Swaroop Darbha
Co-PI: Dr. Sivakumar Rathinam
Department of Mechanical Engineering
Texas A & M University, College Station, TX-77843

AFOSR Program: Dynamics and Control

Contents

Chapter 1

Introduction

1.1 Abstract

The project dealt with two classes of core decision-making algorithms related to operator-UV collaboration; the first class involves the routing of UVs through the set of targets nominated by the operator and the second class of problems involves decision-making algorithms for UVs to accommodate uncertainty. We have developed approximation, lower bounding and exact algorithms to address the two classes of problems. We have also implemented these algorithms in simulations to corroborate the performance of these algorithms. In the ensuing discussion, we will summarize our work for the project, and our main results.

1.2 Routing of UVs

1.2.1 Problem Statement:

In the COUNTER project of AFRL, a collection of UVs were required to gather information about a set of potential targets and transmit them to an operator, who may request a revisit of the potential targets from a different perspective. The operator specifies the location of the potential targets through a human-machine interface (Vigilant Spirit Station) just before the mission begins. The role of the operator is more of a classifier/sensor and of a supervisor in the loop and the task of motion planning is automated. The central computer associated with the Vigilant Spirit Station has a few minutes to compute the trajectories before each mission, including specifying the set of targets to be visited and the order in which they must be visited by each UV and provide them with waypoints.

The basic task allocation/routing problem that arises in this scenario is as follows: Given a set S of potential targets to be visited by a group of m heterogeneous UVs, what is a time optimal (or distance optimal) motion schedule for each of the UVs? There may also be other constraints of timing and precedence that must be obeyed in routing the vehicles.

We have focused on three main issues that complicate the routing problem. The first issue arises due to the heterogeneity among vehicles. Heterogeneity can arise due to UVs

having different motion constraints or due to UVs carrying different suites of sensors. Heterogeneity leads to a number of fundamental routing problems, which have not yet been addressed in the literature but have a wide applicability to defense. The second complicating issue deals with determining the approach angle at each target and the sequence of targets for each vehicle to visit simultaneously. This problem can be posed as a non-linear program and prior to our work, there were no exact or lower bounding algorithms for solving the same. The third complicating issue deals with a constraint on the length of the path traveled by the vehicle before refueling. Small UVs, in particular, have constraints on the amount of fuel it can carry. Surveillance missions with small UVs typically require the vehicles to frequently revisit the depots for refueling.

We focused on three types of algorithms for addressing the routing problems in the project:

- Approximation Algorithms: Algorithms that guarantee producing feasible solutions with the bound on the running time to be a polynomial function of the size of the problem and a bound on sub-optimality. Approximation algorithms provide a priori guarantee on the quality of the solution (which can be inferred from the bound on sup-optimality) even before the solution is computed and tends to be conservative. Approximation algorithms provide a feasible solution, which can serve as a starting point for refinement heuristics. Development of an approximation algorithm requires paying attention to the problem formulation, which in turn helped us in obtaining better a posteriori bounds.

- Transformation methods: The basic idea of a transformation method is to convert the given routing problem to the canonical routing problem, the TSP. By doing so, one can utilize the existing algorithms for the TSP to solve the given routing problem. The basic idea of transformation is as follows: Suppose there are m vehicles, each of which must be driven by an operator. The problem of finding a schedule for the UVs can now be seen as a problem of determining the schedule for the operator, which is a TSP.

- Bounds using Linear Programming (LP) Relaxation and heuristics: We focused on lower bounds, because one can then estimate the quality of a feasible solution a posteriori. Since the routing problem has a partitioning problem embedded in it, we used LP relaxations of the mixed-integer LP to construct a partitioning scheme. Subsequently, we used the Lin-Kernighan-Helsgaun (LKH) heuristic to obtain schedules for each UV.

1.2.2 Summary of our work on Heterogeneity:

We have developed the following approximation algorithms for variants of the heterogeneous routing problems. The basic idea here is to solve a tight Linear Programming (LP) relaxation of the problem and assign each target to a vehicle by rounding some of the fractional variables. Once the targets are partitioned, the standard approximation algorithms for a single TSP are used to obtain a tour for each vehicle.

- $1.5m$-approximation ratio for a Multiple Depot, Heterogeneous TSP where m is the number of vehicles (**We discuss this work in Chapter 2**).
- $4m$-approximation ratio for a Multiple Depot, Heterogeneous HPP.
- $5m/3$-approximation ratio for a Multiple Depot, Multiple Terminal, Heterogeneous HPP.

For the special case where each UV is modeled as a ground robot that can travel forwards and backwards with a constraint on its minimum turning radius, we have developed a 2-approximation algorithm using the Primal-Dual method.

Transformation Methods: The idea of the transformation method is to duplicate the depots and identify them as two locations - one where the vehicle starts from and the other to which the vehicle returns (terminal). Then, one may convert the problem of planning the motion of vehicles to the scheduling of the operator for the vehicles so that the total cost is minimized by setting the cost of the operator switching from a terminal to a depot is zero. There are other modifications to the graph one must make so that the optimal solution to the TSP provides an optimal schedule for the vehicles. More details of this scheme may be found in our publications.

1.2.3 Summary of our work on dealing with motion constraints:

We have developed a lower bounding algorithm for the problem of simultaneously determining the approach angle at the targets and the tours for the UVs. A tighter lower bound helps in quickly and accurately estimating the quality of solutions supplied by heuristics and in the development of exact algorithms for finding optimal motion plans through Branch and Bound procedures. The best known lower bound that is currently available for the motion-constrained routing problem relaxes all motion constraints. We obtain a lower bound by relaxing the constraints corresponding to the angle of approach at each of the targets and then penalizing them whenever they are violated. The solution to the Lagrangian relaxation gives a lower bound, and this lower bound is maximized over the penalty variables using subgradient optimization. Simulation results for a single vehicle problem with 50 targets show that our method improved the previously known lower bounds by almost 15%.

1.2.4 Summary of our work on dealing with fuel constraints:

The basic problem here is as follows: Given a collection of assigned targets, a collection of UVs with limited fuel capacity and a set of depots where they can refuel, what is an optimal motion plan for the collection so that the fuel/storage capacity constraints of every UV is not violated at any point during its motion and the total fuel/distance spent is a minimum. A variant of this problem arises in an application supplied by AFRL Dayton where the UVs pick up images and need to drop them off at dropoff nodes that are hardwired to a remote location. The objective here is to route the UVs so that the

latency in delivering information to the dropoff nodes is within an acceptable value while the total fuel consumed is minimized.

Fuel capacity of a UV may be parameterized by the non-dimensional factor, a, which denotes the maximum fraction of fuel capacity expended while visiting a target from any other target. This factor must necessarily be less than 1/2; otherwise, a UV may run out of fuel before returning to a depot. We have developed a two-approximation (primal-dual) algorithm for a heterogeneous collection of UVs with no fuel capacity constraints and a $2(1 + 2a)/(1 - 2a)$ approximation algorithm for UVs with capacity constraints. Also, for the asymmetric case, we have developed an algorithm with an approximation ratio of $(1 + a + 2a\beta)log(n)/(1 - 2a)$ where n represents the number of targets and a, β are data dependent constants (**We discuss this work in Chapter 3**). In addition to the theoretical guarantees, the following are the contributions of our work:

1. We have developed two mixed-integer linear programs for the routing problem based on the single and multi-commodity flow formulations available for standard network synthesis problems in the literature. These formulations are mainly used to find the optimal solutions for the routing problem, and for corroborating the quality of the solutions produced by the heuristics.

2. Fast construction and improvement heuristics were developed to find good feasible solutions for the routing problem. Even though the mixed integer, linear programming formulations can be used to find optimal solutions, it may be time consuming to solve them. In addition, practical scenarios may only provide approximate input data about the locations of the targets, and as a result, it may be useful to find good, approximate solutions than find optimal solutions that are more difficult to solve. For this reason, we have developed several improvement heuristics for the routing problem with fuel constraints.

3. We have implemented the proposed algorithms and the computational results show that solutions whose costs are, on an average, within 1.5% of the optimum can be obtained relatively fast for a single vehicle problem involving 25 targets.

1.3 Stochastic Dynamic Programming

Problem Statement: We focused on the development and analysis of sub-optimal decision algorithms for a collection of robots that assist a remotely located operator in perimeter surveillance. The operator is tasked with the classification of an incursion across the perimeter. Whenever there is an incursion into the perimeter, an Unattended Ground Sensor (UGS) in the vicinity, signals an alert. A robot services the alert by visiting the alert location, collecting evidence in the form of video and other imagery, and transmitting them to the operator.

The accuracy of operator's classification depends on the volume and freshness of information gathered and provided by the robots at locations where incursions occur. There are two competing needs for a robot: it needs to spend adequate time at an alert location to collect evidence for aiding the operator in accurate classification but it also

needs to service the alert as soon as possible, so that the evidence collected is relevant. The control problem is to determine the optimal amount of time a robot must spend servicing an alert. The incursions are stochastic and their statistics is known a priori.

1.3.1 Summary of results:

This problem may be posed as a Markov Decision Problem (MDP). However, even for two robots and five UGS locations, the number of states is of the order of millions. We adopted Approximate Dynamic Programming (ADP) via Linear Programming (LP) as it provides a way to approximate the value function and provide bounds on its sub-optimality. The novel feature of this work is to present a lower bound via LP based techniques and state partitioning and construct a sub-optimal policy whose performance exceeds the lower bound.

Specifically, we obtained the following results:

1. We developed upper and lower bounds to the value function as component-wise minimum vector of all feasible solutions to Generalized Bellman inequalities (GBIs). These inequalities require the specification of disjoint sets of the state space and the bounds take a constant value for all the states in specified disjoint sets. The bounds help establish sub-optimality bounds for any feasible stationary policy for the MDP. **We discuss this work in Chapter 4.**

2. We developed a sub-optimal policy, which is greedy with respect to the constructed lower bound and show that its performance is no less than the lower bound.

3. We exploit the structure of the perimeter surveillance problem and simplify the computation of the upper and lower bounds to the determination of optimal solution of a linear program or a sequence of linear programs in fewer variables.

4. We constructed a sub-optimal policy, which was implemented by AFRL researchers at Vandenberghe AFB.

Acknowledgment/Disclaimer: This work was sponsored (in part) by the Air Force Office of Scientific Research under grant/contract number FA09550-10-1-0932. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research or the US government.

1.4 Graduate Students Supported using the Grant

1. Myoungkuk Park ME Graduate Student, Texas A&M University. 2. Manyam Gupta ME Graduate Student, Texas A&M University. 3. Kaarthik Sundar ME Graduate Student, Texas A&M University. 4. Jungyun Bae ME Graduate Student, Texas A&M University.

1.5 Publications:

1. P. Oberlin, S. Rathinam and S. Darbha, "Today's Traveling Salesman Problem," IEEE Robotics and Automation Magazine, vol. 17, no. 4, pp.70-77, December 2010.

2. S. Yadlapalli, S. Rathinam and S. Darbha, "A constant factor approximation algorithm for a multiple depot, heterogeneous Traveling Salesmen Problem," accepted for publication in Optimization Letters, October 2010.

3. R. Doshi, S. Yadlapalli, S. Rathinam, and S. Darbha, "Approximation algorithms and heuristics for a 2-depot Heterogeneous Hamiltonian Path Problem," accepted for publication in the International Journal of Robust Nonlinear Control, November 2010.

4. K. Krishnamoorty, M. Pachter, P. R. Chandler and S. Darbha, "Approximate Dynamic Programming with State Aggregation applied to UV Perimeter Patrol," accepted for publication in the International Journal of Robust Nonlinear Control, November 2010.

5. S. Darbha, K. Krishnamoorthy, M. Pachter and P. Chandler, "State Aggregation based Linear Programming Approach to Approximate Dynamic Programming," Proceedings of the 49th IEEE Conference on Decision and Control, pp. 935-941, December 2010.

6. Sai Krishna Yadlapalli, Jung Yun Bae, Sivakumar Rathinam, Swaroop Darbha, "Approximation Algorithms for Variants of a Heterogenous Multiple Depot Hamiltonian Path Problem", to be presented at the American Control Conference, June 2011.

7. Krishnamoorthy Kalyanam, Meir Pachter, Phillip R. Chandler, David W. Casbeer, Swaroop Darbha, "UV Perimeter Patrol Operations Optimization Using Efficient Dynamic Programming", to be presented at the American Control Conference, June 2011.

8. K. Sundar and S. Rathinam, "Route Planning Algorithms for Unmanned Aerial Vehicles with Refueling Constraints", Accepted in the IEEE Transactions on Automation Science and Engineering, 2013.

9. K. Sundar and S. Rathinam, "A Primal-Dual Heuristic for a Heterogeneous Unmanned Vehicle Path Planning Problem", Accepted in the International Journal of Advanced Robotic Systems, 2013.

10. K. Krishnamoorthy, M. Pachter, S. Darbha and P. Chandler, "Optimization of Perimeter Patrol Operations using UVs", AIAA J. Guidance, Control and Dynamics, 35(2): 434-441, 2012.

11. K. Krishnamoorthy, M. Pachter, S. Darbha and P. Chandler, "ADP with State Aggregation applied to UV Perimeter Patrol", Intl. J. Robust and Nonlinear Control, 21(12): 1396-1409.

12. K. J. Obermeyer, P. Oberlin, and S. Darbha, "Sampling-Based Path Planning for a Visual Reconnaissance UV", AIAA Journal of Guidance, Control, and Dynamics, 2012.

13. M. Park, K. Krishnamoorthy, S. Darbha, P. Chandler and M. Pachter, "State partitioning based linear program for stochastic dynamic programs: An invariance property", Operations Research Letters, Vol. 40, no. 6, November 2012.

14. K. Kalyanam, M. Park, S. Darbha, P. Chandler, M. Pachter, "A Lower Bounding Linear Program for the Perimeter Patrol Optimization Problem", accepted for publication in AIAA Journal of Guidance, Dynamics and Control, April 2013.

15. K. Krishnamoorthy, M. Park, S. Darbha, M. Pachter and P. Chandler, "Approximate Dynamic Programming Applied to UAV Perimeter Patrol", Recent Advances in Research on Unmanned Aerial Vehicles, Lecture Notes in Control and Information Sciences, Vol. 444, pp. 119-146, 2013.

1.6 Honors and Awards Received:

Semi-plenary speaker at the 2012 ASME Dynamic Systems and Control Conference presenting the paper: "Sub-Optimal Stationary Policies For A Class of Stochastic Optimization Problems Arising in Robotic Surveillance Applications," co-authored by M. Park, K. Krishnamoorty, S. Darbha, P. P. Khargonekar, P. Chandler and M. Pachter.

AFRL Point of Contact Dr. Corey Schumacher, AFRL/RBCA, WPAFB, OH, Phone: 937-255-8682.

Transitions The stochastic control algorithms for perimeter surveillance and inspection problems developed in collaboration with AFRL researchers have been incorporated into the UV software/flight tests by the AFRL researchers. The point of contact is Dr. Corey Schumacher, AFRL/RBCA, WPAFB, OH, Phone: 937-255-8682.

New Discoveries None.

Chapter 2

Approximation Algorithm for a Multiple, Heterogeneous Traveling Salesman Problem

2.1 Introduction

Surveillance applications involving Unmanned Aerial Vehicles (UAVs) or ground robots require multiple vehicles with different capabilities to visit a set of destinations. This chapter addresses an important routing problem that arises in these applications involving two heterogenous vehicles. Specifically, the routing problem we address is a 2-depot, Heterogenous Traveling Salesman Problem (2-HTSP) which is stated as follows: Given a set of destinations and two heterogeneous vehicles that start from distinct depots, find a tour for each vehicle such that each destination is visited exactly once and the total cost of the tours of the vehicles is a minimum.

In this chapter, we consider two types of heterogeneity for both the vehicles, i.e., *structural* heterogeneity and *functional* heterogeneity. If the vehicles are structurally different, the cost of traveling between two destinations not only depends on the position of the destinations but also on the vehicle. In the case of functional heterogeneity, the vehicles are identical structurally but there may be additional vehicle-destination constraints that must be met. In this case, the destinations may be partitioned into three disjoint subsets: a subset of destinations the first vehicle must visit, a subset of destinations the second vehicle must visit and a set of common destinations that either of the two vehicles can visit. Although the cost of traveling from one destination to another is the same for both the vehicles, these restrictions on the vehicle-destination assignment introduce heterogeneity.

There are several applications where routing problems such as the 2-HTSP could arise. In UAV applications, it is possible that the vehicles have different constraints on their maximum speeds depending on the vehicle type. Even if we ignore the constraints on the turning radius of the vehicles when the destinations are reasonably far apart, the cost of traveling between any two destinations is still dependent on the type of the vehicle. Also, the UAVs can carry different sensors, and therefore, there may be additional constraints

that require a subset of destinations must be visited by a specific UAV.

The 2-HTSP is a generalization of the single Traveling Salesman Problem and is NP-Hard (1). Therefore, we are interested in developing approximation algorithms for the 2-HTSP. An $\alpha-$approximation algorithm (1) is an algorithm that

- has a polynomial-time running time, and
- returns a solution whose cost is within α times the optimal cost for any instance of the problem.

We will assume that the cost of traveling from an origin to a destination directly for each vehicle is no more expensive than the cost of traveling from the same origin to the destination through an intermediate location. We say that the costs satisfy the triangle inequality if they satisfy the above property. It is currently known that there cannot exist a constant factor approximation algorithm for a single Traveling Salesman Problem in general if the triangle inequality is not satisfied unless $P = NP$. In this chapter, we present a 3-approximation algorithm for the 2-HTSP when the costs associated with each vehicle satisfy the triangle inequality.

2.2 Literature Review

The 2-HTSP is related to a well known class of problems that has received significant attention in the area of combinatorial optimization. These problems include the Traveling Salesman Problem (TSP), the Hamiltonian Path Problem (HPP) and their generalizations. As this work deals with constant factor approximation algorithms, henceforth, we assume that, for every vehicle the costs satisfy the triangle inequality. The symmetric TSP has two well known approximation algorithms - the $2-$approximation algorithm obtained by doubling the minimum spanning tree (MST) and the $1.5-$approximation algorithm of Christofides obtained through the construction of MST and a weighted non-bipartite matching of nodes of MST with odd degree (2).

There are $2-$approximation algorithms for variants of the homogenous, multiple TSP and HPP in (3),(4). Prior to our work, there were no approximation algorithms for a general heterogenous, multiple TSP in the literature. In this work, we present the first 3-approximation algorithm for the 2-HTSP when the costs satisfy the triangle inequality.

Contributions of our work: We formulate the 2-HTSP as an integer program with assignment, degree and connectivity constraints on a multi-graph. Given any two destinations, we construct this multi-graph by adding an edge joining the two destinations for each vehicle. The cost assigned to an edge would then be equal to the distance required by the corresponding vehicle to travel that edge. We use a set of binary decision variables to formulate the assignment constraints between the vehicles and the destinations. As each destination must be visited exactly once, we also have a degree constraint on each destination vertex. If a destination is assigned to a vehicle, the connectivity constraints require that there must be at least two edge-disjoint paths between the destination and the depot corresponding to the vehicle. Using Menger's theorem(1), this connectivity requirement can then be formulated as a cut constraint. To formulate these cut and the

degree constraints, we use binary decision variables to decide whether an edge must be chosen or not.

The basic idea for the approximation algorithm is as follows: We first relax all the binary decision variables and solve the resulting linear program to find the subset of destinations each vehicle must visit. Once the partitioning problem is solved, Christofides algorithm (2) is used on the partitions to get a tour for each vehicle. Even though the number of cut constraints in this linear program grow exponentially with the number of destinations, the relaxed linear program can be shown to be solvable in polynomial time using the Ellipsoid method(5). Using the result that the cost of the feasible solution produced by the Christofides algorithm is at most $\frac{3}{2}$ times the cost of the Held-Karp relaxation of the single TSP (6) and the parsimonious property of the Held-Karp relaxation(7), one can show that the proposed algorithm has an approximation ratio of 3. A key part of our approximation algorithm is in the way we formulate the 2-HTSP and relax the constraints. We present this formulation in the next section.

2.3 Problem Formulation

Let $T = \{1, \ldots, n\}$ be the set of vertices that denote all the destinations and $V = \{d_1, d_2\}$ be the set of vertices that correspond to the initial depots of the vehicles. For each depot vertex d_i, we also introduce a copy of the depot vertex called the terminal vertex, d_i', that exactly coincides with the location of the depot vertex. Each vehicle after visiting its share of destinations will visit its corresponding terminal before returning to its depot. Our integer programming formulation includes a terminal vertex for each vehicle in order to allow for each vehicle to visit exactly one destination if needed (this will be further discussed in the remarks later in this section).

Let $\mathtt{V}^i = \{d_i, d_i'\} \bigcup T$ denote the set of all the vertices corresponding to the i^{th} vehicle. Let E^i stand for the set of all the edges joining any two vertices in $\mathtt{V}^i$. For any $S \subset \mathtt{V}^i$, let $\delta_i(S)$ denote the set of all the edges $e \in E^i$ that has one end point in S and one end point in $\mathtt{V}^i \setminus S$. Each edge $e \in E^i$ has a cost $C_e^i \in \mathbb{Q}^+$ associated with it where $\mathbb{Q}^+$ is the set of all positive rational numbers. Assume that all the costs satisfy the triangle inequality. Let R_1 and R_2 be the set of vertices that must be visited by the first and the second vehicle respectively. Note that $R_1 \bigcap R_2 = \emptyset$ and each destination in $T \setminus (R_1 \bigcup R_2)$ can be visited by either the first or the second vehicle.

Let x_e (y_e) denote the binary variable that decide whether edge e is present in the routes of the first (second) vehicle. An edge e is present in the tour of the first vehicle if $x_e = 1$ and is not present otherwise. y_e is defined similarly. Let ϕ_i denote the binary decision variable that is equal to 1 if destination i is visited by the first vehicle and is equal to 0 otherwise. Similarly, let η_i denote the binary decision variable that is equal to 1 if destination i is visited by the second vehicle and is equal to 0 otherwise. The following is the **integer programming formulation** of the 2-HTSP:

$$
\begin{aligned}
C_{opt} &= \min_{x,y,\phi,\eta} \sum_{e\in E^1} x_e C_e^1 + \sum_{e\in E^2} y_e C_e^2 && (2.1)\\
\phi_i &= 1, \text{ for all } i \in R_1, && (2.2)\\
\eta_i &= 1, \text{ for all } i \in R_2, && (2.3)\\
\phi_i + \eta_i &= 1, \text{ for all } i \in T \setminus (R_1 \bigcup R_2), && (2.4)
\end{aligned}
$$

$$
\begin{aligned}
\sum_{e\in\delta_1(\{i\})} x_e &= 2\phi_i, \forall i \in T, && (2.5a)\\
\sum_{e\in\delta_1(\{u\})} x_e &\geq 2\phi_i, u \in \{d_1, d_1'\}, \forall i \in T, && (2.5b)\\
\sum_{e\in\delta_1(\{u\})} x_e &\leq 2, u \in \{d_1, d_1'\}, && (2.5c)
\end{aligned}
$$

For all $i \in T$,

$$
\begin{aligned}
\sum_{e\in\delta_1(S)} x_e &\geq 2\phi_i, \forall S \subset \mathtt{V}^1, \text{ such that}\\
&\quad i \in S, \ |S \bigcap \{d_1, d_1'\}| \leq 1, && (2.5d)\\
x_e &\in \{0,1\} \ \forall e \in E^1, && (2.5e)\\
\phi_i &\in \{0,1\} \ \forall i \in T. && (2.5f)
\end{aligned}
$$

$$
\begin{aligned}
\sum_{e\in\delta_2(\{i\})} y_e &= 2\eta_i, \forall i \in T, && (2.6a)\\
\sum_{e\in\delta_2(\{u\})} y_e &\geq 2\eta_i, u \in \{d_2, d_2'\}, \forall i \in T, && (2.6b)\\
\sum_{e\in\delta_2(\{u\})} y_e &\leq 2, u \in \{d_2, d_2'\}, && (2.6c)
\end{aligned}
$$

For all $i \in T$,

$$
\begin{aligned}
\sum_{e\in\delta_2(S)} y_e &\geq 2\eta_i, \forall S \subset \mathtt{V}^2, \text{ such that}\\
&\quad i \in S, \ |S \bigcap \{d_2, d_2'\}| \leq 1, && (2.6d)\\
y_e &\in \{0,1\} \ \forall e \in E^2, && (2.6e)\\
\eta_i &\in \{0,1\} \ \forall i \in T. && (2.6f)
\end{aligned}
$$

The constraints in equations (2.2) and (2.3) state that the destinations in R_1 and R_2 must be respectively visited by the first and the second vehicle. The assignment constraints in equation (2.4) require that a destination in $T \setminus (R_1 \bigcup R_2)$ can be visited either by the first vehicle or the second vehicle but not both. The degree constraints in equations (2.5a, 2.6a) together indicate that the number of edges incident on each destination vertex must be equal to 2. The degree constraints in equations (2.5b,2.6b) specify that the number of edges incident on a depot/terminal must be at least equal to 2 if the vehicle corresponding to the depot/terminal is visiting at least one destination. The degree constraints in equations (2.5c,2.6c) state that the number of edges incident on both the depots and the terminals can at most be equal to 2. If a destination is visited by a vehicle, the cut constraints in equations (2.5d, 2.6d) enforce a requirement that there must be at least two edge disjoint paths from the destination to the depot/terminal corresponding to the vehicle visiting that destination. These cut constraints in combination with the degree constraints also eliminate the presence of any cycles among the destination vertices.

Remark 1 *The terminal vertices d'_1, d'_2 were added to the problem to essentially allow for a vehicle to visit exactly one destination if needed. For example, by adding these terminal vertices, one could allow a tour for the first vehicle to be of the form $\{d_1, u, d'_1, d_1\}$ where u is a vertex denoting a destination. Then, the first vehicle visits the destination u, and then the terminal d'_1 before returning to its depot. However, adding these terminal vertices could also result in a solution where the optimal tour for the i^{th} vehicle is of the form $\{d_i, v_{i1}, \cdots, v_{il_i}, d'_i, v_{il_{i+1}}, \cdots, v_{ik_i}, d_i\}$ where $v_{ij} \in T$ for $j = 1, \cdots, k_i$. In this case, the depot and its corresponding terminal vertex are not adjacent vertices in the tour. However, this is not an issue in this chapter as we assume that all the costs associated with every vehicle satisfy the triangle inequality. Therefore, one can always shortcut the edges in the optimal solution to obtain tours so that each vehicle returns to its depot immediately after visiting its corresponding terminal.*

Remark 2 *The cut constraints in equations (2.5d,2.6d) can also be written equivalently as given below. The reason for formulating the constraints as stated in equations (2.5d,2.6d) is to simplify the proofs of the approximation algorithm discussed in section 2.5.*

$$\sum_{e \in \delta_1(S)} x_e \geq 2 \max_{i \in S} \phi_i, \forall S \subset \mathtt{V}^1 \text{ such that } |S \bigcap T| > 0, \ |S \bigcap \{d_1, d'_1\}| \leq 1,$$

$$\sum_{e \in \delta_2(S)} y_e \geq 2 \max_{i \in S} \eta_i, \forall S \subset \mathtt{V}^2 \text{ such that } |S \bigcap T| > 0, \ |S \bigcap \{d_2, d'_2\}| \leq 1.$$

Remark 3 *Using the max-flow min-cut theorem (1), the cut constraints in equations (2.5d,2.6d) can also be formulated using flow constraints. Therefore, ϕ_i and η_i can be interpreted as the amount of flow shipped from the first and second depot respectively to the i^{th} destination.*

Before we present the approximation algorithm, we state the following Linear Programming (LP) relaxation of the 2-HTSP as it plays a crucial role in the development of

the algorithm.

$$C^*_{lp} = \min_{x,y,\phi,\eta} \sum_{e\in E^1} x_e C^1_e + \sum_{e\in E^2} y_e C^2_e \quad (2.7)$$

$$\phi_i \geq 1, \text{ for all } i \in R_1, \quad (2.8)$$

$$\eta_i \geq 1, \text{ for all } i \in R_2, \quad (2.9)$$

$$\phi_i + \eta_i \geq 1, \text{ for all } i \in T \setminus (R_1 \bigcup R_2), \quad (2.10)$$

$$\sum_{e\in\delta_1(\{u\})} x_e \geq 2\phi_i, u \in \{d_1, d'_1\}, \quad (2.11a)$$

For all $i \in T$,

$$\sum_{e\in\delta_1(S)} x_e \geq 2\phi_i, \forall S \subset \mathtt{V}^1, \text{ such that } i \in S,\ |S \bigcap \{d_1, d'_1\}| \leq 1, \quad (2.11b)$$

$$0 \leq x_e \leq 1\ \forall e \in E^1, \quad (2.11c)$$

$$\phi_i \geq 0\ \forall i \in T. \quad (2.11d)$$

$$\sum_{e\in\delta_2(\{u\})} y_e \geq 2\eta_i, u \in \{d_2, d'_2\}, \quad (2.12a)$$

For all $i \in T$,

$$\sum_{e\in\delta_2(S)} y_e \geq 2\eta_i, \forall S \subset \mathtt{V}^2, \text{ such that } i \in S,\ |S \bigcap \{d_2, d'_2\}| \leq 1, \quad (2.12b)$$

$$0 \leq y_e \leq 1\ \forall e \in E^2, \quad (2.12c)$$

$$\eta_i \geq 0\ \forall i \in T. \quad (2.12d)$$

2.4 Approximation Algorithm for the 2-HTSP

The following is the proposed algorithm *Approx* for the 2-HTSP:

1. Solve the Linear Programming relaxation formulated in equations (2.7-2.12) using the Ellipsoid method (5). Let an optimal solution to this relaxation be denoted by $(x^*, y^*, \phi^*, \eta^*)$. We will later show that this relaxation is solvable in polynomial time.

2. ϕ^*_i (η^*_i) essentially denotes the optimal fraction of the flow shipped to the i^{th} destination using the first vehicle (second vehicle). Assign each destination to the vehicle that ships its largest fraction. Break ties arbitrarily. This step of the algorithm essentially partitions the destinations into two disjoint groups. Let $\mathcal{U}_1 = \{i : i \in T, \psi_i \geq \eta_i\}$ correspond to those destinations which are assigned to the first vehicle, and $\mathcal{U}_2 = T \setminus \mathcal{U}_1$ be the set of destinations assigned to the second vehicle.

3. For the i^{th} vehicle, if $\mathcal{U}_i$ is not empty, apply the Christofides algorithm to find a tour that visits all the vertices in $\mathcal{U}_i \bigcup \{d_i, d'_i\}$.

Clearly, the tours produced by the above algorithm is a feasible solution for the integer program formulated in equations (2.1-2.6f). The following theorem is the main result of this work:

Theorem 2.4.1 *Algorithm Approx is a polynomial time algorithm for the 2-HTSP with an approximation ratio of 3.*

2.5 Proof of the 3-approximation ratio of *Approx*

In the following lemma, we first show that *Approx* is a polynomial time algorithm.

Lemma 2.5.1 *Approx is a polynomial time algorithm.*

Proof The main steps in *Approx* involve solving a linear program defined by equations (2.7-2.12) and using the Christofides algorithm. If there are n destinations, it is known that the number of steps required for the Christofides algorithm is of $O(n^3)$. Therefore, step (3) of the algorithm *Approx* requires $O(|\mathcal{U}_1|^3) + O(|\mathcal{U}_2|^3) \approx O(n^3)$ steps. We now show that the linear program (2.7-2.12) is solvable in polynomial time using the Ellipsoid method (5). In (5), Grötschel, Lovász and Schrijver showed that the polynomial solvability of a linear program is equivalent to the polynomial solvability of the following separation problem using the Ellipsoid method:

Let $\mathcal{P}$ denote the polytope defined by all the constraints of the linear program in equations (2.8-2.12). Given $x_e \; \forall e \in E^1, y_e \; \forall e \in E^2$, and $\phi_i, \eta_i \; \forall i \in T$, decide whether the given solution is in $\mathcal{P}$ and if not, find a violated constraint.

The cut constraints defined by equations (2.11b,2.12b) are the only set of constraints that grow exponentially with the number of destinations. Therefore, the separation problem is solvable in polynomial time if a separation algorithm can be developed for these cut constraints. For each destination $i \in T$, the cut constraints defined in equation (2.11b) are as follows:

$$\sum_{e \in \delta_1(S)} x_e \;\; \geq \;\; 2\phi_i, \forall S \subset \mathtt{V}^1 \text{ such that } i \in S \text{ and } |S \bigcap \{d_1, d_1'\}| \leq 1. \qquad (2.13)$$

Applying max-flow, min-cut theorem, the above cut constraints imply that there must at least be a flow of $2\phi_i$ from vertex i to both the depot d_1 and the terminal d_1'. Therefore, given a destination vertex $i \in T$, $x_e \; \forall e \in E^1$ and ϕ_i, one can use the max-flow algorithm to decide whether the given solution is feasible for the constraints in equation (2.13) or find a cut that violates these constraints in polynomial time. By repeating this argument for each of the destination vertices, we can conclude that a polynomial time separation algorithm is available to handle the constraints defined in equation (2.11b). By using similar arguments, one can also develop a separation algorithm for the constraints in equation (2.12b). Therefore, there is a polynomial time algorithm for the separation problem. Hence, the linear program defined in equations (2.11,2.12) is solvable in polynomial time using the Ellipsoidal method (5).

In the remaining part of this discussion, we will show that the approximation ratio of *Approx* is 3. Let the tour produced for the i^{th} vehicle by *Approx* be denoted by $TOUR_i$. Let the cost of these tours be denoted by $C(TOUR_1)$ and $C(TOUR_2)$ respectively. For

a single TSP, Shmoys and Williamson (6) have shown that the cost of the solution produced by the Christofides algorithm is at most a factor of $\frac{3}{2}$ away from the optimal cost of the Held-Karp relaxation of the single TSP. Using this result, one can deduce that $C(TOUR_1) \leq \frac{3}{2}C^1_{hk}$ where C^1_{hk} denotes the optimal cost of the Held-Karp's relaxation for the first vehicle visiting all the vertices in $\mathcal{U}_1 \bigcup \{d_1, d'_1\}$. Similarly, it follows that $C(TOUR_2) \leq \frac{3}{2}C^2_{hk}$ where C^2_{hk} denotes the optimal cost of the Held-Karp's relaxation for the second vehicle visiting all the vertices in $\mathcal{U}_2 \bigcup \{d_2, d'_2\}$. The relaxation costs C^1_{hk} and C^2_{hk} are essentially defined as follows:

$$
\begin{aligned}
C^1_{hk} &= \min_x \sum_{e \in E^1} x_e C^1_e \\
\sum_{e \in \delta_1(S)} x_e &\geq 2, \forall S \subset \mathcal{U}_1 \bigcup \{d_1, d'_1\}, \\
\sum_{e \in \delta_1(\{i\})} x_e &= 2, \forall i \in \mathcal{U}_1 \bigcup \{d_1, d'_1\}, \\
\sum_{e \in \delta_1(\{i\})} x_e &= 0, \forall i \in \mathcal{U}_2, \\
x_e &\geq 0 \ \forall e \in E^1.
\end{aligned}
\qquad
\begin{aligned}
C^2_{hk} &= \min_y \sum_{e \in E^2} y_e C^2_e \\
\sum_{e \in \delta_2(S)} y_e &\geq 2, \forall S \subset \mathcal{U}_2 \bigcup \{d_2, d'_2\}, \\
\sum_{e \in \delta_2(\{i\})} y_e &= 2, \forall i \in \mathcal{U}_2 \bigcup \{d_2, d'_2\}, \\
\sum_{e \in \delta_2(\{i\})} y_e &= 0, \forall i \in \mathcal{U}_1, \\
y_e &\geq 0 \ \forall e \in E^2.
\end{aligned}
$$

As all the costs satisfy the triangle inequality, Goemans and Bertsimas (7) have shown that the optimal relaxation cost will not change if one were to remove all the degree constraints in the above Held-Karp relaxation. In (7), Goemans and Bertsimas proved this property for a more general survivable network design problem. This property is essentially called the parsimonious property of a network design problem. That is,

$$
\begin{aligned}
C^1_{hk} &= \min_x \sum_{e \in E^1} x_e C^1_e \qquad (2.14) \\
\sum_{e \in \delta_1(S)} x_e &\geq 2, \forall S \subset \mathcal{U}_1 \bigcup \{d_1, d'_1\}, \\
x_e &\geq 0 \ \forall e \in E^1.
\end{aligned}
\qquad
\begin{aligned}
C^2_{hk} &= \min_y \sum_{e \in E^2} y_e C^2_e \qquad (2.15) \\
\sum_{e \in \delta_2(S)} y_e &\geq 2, \forall S \subset \mathcal{U}_2 \bigcup \{d_2, d'_2\}, \\
y_e &\geq 0 \ \forall e \in E^2.
\end{aligned}
$$

The sum of the optimal cost of the Held-Karp relaxations, $C^1_{hk} + C^2_{hk}$, can now be upper bounded by two times the optimal cost, C^*_{lp}, of the LP relaxation (equations 2.7-2.12) of the 2-HTSP. To prove this, consider any optimal solution $(x^*, y^*, \phi^*, \eta^*)$ to the LP in equations (2.7-2.12). One can construct a solution, $\widehat{x}$, for the Held-Karp relaxation in (2.14) by choosing $\widehat{x} = 2x^*$. To prove that $\widehat{x}$ is feasible solution for (2.14), note that,

for any $S \subset \mathcal{U}_1 \bigcup \{d_1, d_1'\}, |S \bigcap \{d_1, d_1'\}| = 2$,

$$\begin{aligned}
\sum_{e \in \delta_1(S)} \widehat{x}_e &= 2 \sum_{e \in \delta_1(S)} x_e^*, \\
&= 2 \sum_{e \in \delta_1(\mathtt{V}^1 \setminus S)} x_e^*, \\
&\geq 4\phi_i^*, \text{ for all } i \in \mathtt{V}^1 \setminus S, \quad \text{(from constraint 2.11}b\text{)} \\
&\geq 4\phi_i^*, \text{ for all } i \in \mathcal{U}_1 \setminus S, \\
&\geq 2.
\end{aligned}$$

Similarly, for any $S \subset \mathcal{U}_1 \bigcup \{d_1, d_1'\}, |S \bigcap \mathcal{U}_1| \geq 1, |S \bigcap \{d_1, d_1'\}| \leq 1$,

$$\begin{aligned}
\sum_{e \in \delta_1(S)} \widehat{x}_e &= 2 \sum_{e \in \delta_1(S)} x_e^*, \\
&\geq 4\phi_i^*, \text{ for all } i \in S \bigcap \mathcal{U}_1, \quad \text{(from constraint 2.11}b\text{)} \\
&\geq 2.
\end{aligned}$$

Also, for $u = d_1$ or $u = d_1'$,

$$\begin{aligned}
\sum_{e \in \delta_1(u)} \widehat{x}_e &= 2 \sum_{e \in \delta_1(u)} x_e^*, \\
&\geq 4\phi_i^*, \text{ for all } i \in \mathcal{U}_1, \quad \text{(from constraint 2.11}a\text{)} \\
&\geq 2.
\end{aligned}$$

Therefore, $\widehat{x}$ is a feasible solution for the Held-Karp relaxation in (2.14). In the same way, one can also show that $\widehat{y} = 2y^*$ is also a feasible solution for the Help-Karp relaxation defined in (2.15). Therefore, $C_{hk}^1 + C_{hk}^2 \leq 2 \sum_{e \in E^1} x_e^* C_e^1 + 2 \sum_{e \in E^2} y_e^* C_e^2 = 2C_{lp}^*$. Putting together all the results, we have

$$\begin{aligned}
C(TOUR_1) + C(TOUR_2) &\leq \frac{3}{2}(C_{hk}^1 + C_{hk}^2), \\
&\leq 3C_{lp}^*, \\
&\leq 3C_{opt}.
\end{aligned}$$

2.6 Extension to the related min-max problem

The above approach can also be extended to obtain a 3-approximation algorithm for a 2 depot, Heterogeneous TSP where the objective is to minimize the maximum cost traveled by either of the vehicles. To see this, consider the following min-max problem:

$$C_{opt}^{max*} = \min_{x,y,\phi,\eta} \max\{\sum_{e \in E^1} x_e C_e^1, \sum_{e \in E^2} y_e C_e^2\} \tag{2.16}$$

subject to the constraints defined in equations (2.2-2.6f). The above min-max problem can also be restated as:

$$C_{opt}^{max*} = \min_{t,x,y,\phi,\eta} t \tag{2.17}$$

$$\begin{aligned} t &\geq \sum_{e \in E^1} x_e C_e^1, \\ t &\geq \sum_{e \in E^2} y_e C_e^2, \end{aligned} \tag{2.18}$$

and the constraints in equations (2.2-2.6f). Therefore, a LP relaxation of this min-max problem will have an objective defined in equation (2.17) subject to constraints in equations (2.18,2.8-2.12). The approximation algorithm for the min-max problem also follows the same approach as Algorithm *Approx* in section 2.4: 1) Solve the LP relaxation of the min-max problem; 2) Assign any destination i to the first vehicle if $\phi_i \geq \eta_i$ and the remaining destinations to the second vehicle; 3) For each vehicle, use the Christofides algorithm to obtain a tour to visit its set of destinations. Let C_{lp}^{max*} be the optimal cost of the LP relaxation of the min-max problem. Using the same notations and similar arguments as in the previous section, we have,

$$\begin{aligned} \max(C(TOUR_1), C(TOUR_2)) &\leq \frac{3}{2} \max(C_{hk}^1, C_{hk}^2), \\ &\leq 3C_{lp}^{max*}, \\ &\leq 3C_{opt}^{max*}. \end{aligned}$$

2.7 Conclusions

In general, the approach given in this work can be extended to obtain a $\frac{3m}{2}$-approximation algorithm for variants of a m-depot, Heterogeneous Traveling Salesman Problem. When there are more than 2 vehicles, the vehicle-destination constraints that are present due to the functional heterogeneity of vehicles can be specified in a couple of different ways. For example, one can specify that a vehicle must visit a subset of destinations or specify that a vehicle must not visit a subset of destinations. The $\frac{3m}{2}$-approximation algorithm that one can obtain by extending the approach in this work to the multiple vehicle case can handle both these specifications.

Chapter 3

Approximation Algorithm for a Routing Problem with Fuel Constraints

3.1 Introduction

Path planning for small Unmanned Aerial Vehicles (UAVs) is one of the research areas that has received significant attention in the last decade. Small UAVs have already been field tested in civilian applications such as wild-fire management (8), weather and hurricane monitoring (9; 10; 11), and pollutant estimation (12) where the vehicles are used to collect relevant sensor information and transmit the information to the ground (control) stations for further processing. Compared to large UAVs, small UAVs are relatively easier to operate and are significantly cheaper. Small UAVs can fly at low altitudes and can avoid obstacles or threats at low altitudes more easily. Even in military applications, small vehicles (13) are used frequently for intelligence gathering and damage assessment as they are easier to fly and can be hand launched by an individual without any reliance on a runway or a specific type of terrain.

Even though there are several advantages with using small platforms, they also come with other resource constraints due to their size and limited payload. As small UAVs typically have fuel constraints, it may not be possible for an UAV to complete a surveillance mission before refueling at one of the depots. For example, consider a typical surveillance mission where a vehicle starts at a depot and is required to visit a set of targets. To complete this mission, the vehicle might have to start at the depot, visit a subset of targets and then reach one of the depots for refueling before starting a new path. One can reasonably assume that once the UAV reaches a depot, it will be refueled to full capacity before it leaves again for visiting any remaining targets. If the goal is to visit each of the given targets at least once, then the UAV may have to repeatedly visit some depots in order to refuel again before visiting all the targets. In this scenario, the following **Fuel Constrained, UAV Routing Problem** (FCURP) naturally arises: Given a set of targets and depots, and an UAV where the vehicle is initially stationed at one of the depots, find a path for the UAV such that each target is visited at least once, the fuel

constraint is never violated along the path for the UAV, and the travel cost for the vehicle is a minimum. The travel cost is defined as the total fuel consumed by the vehicle as it traverses its path. We assume here that the fuel required to travel a given path for the UAV is directly proportional to the length of the path. Please refer to figure 3.1 for an illustration of this problem.

The main difficulty with the FCURP is purely combinatorial. As long as a path of minimum length can be efficiently computed from an origin to a destination for the UAV, the motion constraints of the UAVs do not complicate the problem. To emphasize this point, let the UAV be modeled as a Dubins' (14) vehicle. If the optimal heading angle is specified at each target, the problem of finding the optimal sequence of targets to be visited reduces to a generalization of the TSP, which is known to be NP-Hard (1). However, if the optimal sequence were to be given, the optimal heading angles can be found using Dynamic Programming and can be determined arbitrarily accurately using efficient algorithms as shown in (15). We only address the combinatorial difficulty in this problem and assume that the UAV must visit each target at a specified heading angle. As a result, the travel costs for the UAV may be asymmetric. Asymmetry means that the cost of traveling the optimal path from target A with heading ψ_A and arriving at target B with heading ψ_B may not equal the length of its optimal path starting at target B with heading ψ_B and arriving at target A with heading ψ_A.

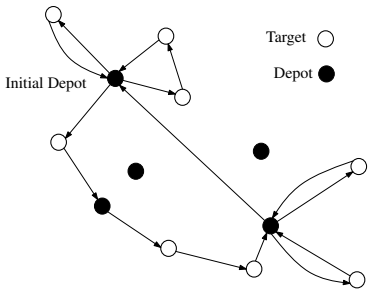


Figure 3.1: A possible path for the UAV which visits all the targets while visiting some depots for refueling. Note that a depot can be visited any number of times for refueling while some depots may not be visited at all.

The FCURP specifically arises in Intelligence, Surveillance and Reconnaissance (ISR) missions such as the Cooperative Operations in Urban Terrain (COUNTER) project at the Air Force Research Laboratory (16; 17). In this project, an UAV or a team of UAVs with limited payload and fuel constraints are used to gather information about a set of potential targets. The operator specifies the location of targets through a human-machine interface, and the central computer associated with this interface is required to develop a trajectory for each UAV in a few minutes. Here, the computer plans the mission and the UAVs spend their limited resources in collecting and transmitting the information about the targets to the ground station. Our goal is to develop fast algorithms so that the central computer can find good feasible solutions to routing problems such as the FCURP as quickly as possible. We address this goal through the development of an approximation algorithm and heuristics in this chapter.

A α-approximation algorithm for an optimization problem is an algorithm that runs

in polynomial time and finds a feasible solution whose cost is at most α times the optimal cost for every instance of the problem. This guarantee α is also referred to as the approximation factor of the algorithm. This approximation factor provides a theoretical upper bound on the quality of the solution produced by the algorithm for any instance of the problem. These upper bounds are known a-priori, *i.e.*, they are known even before one implements the approximation algorithm for some specific instances of the problem. For these reasons, the bound provided by the approximation factor is generally conservative.

Currently, there are no constant factor approximation algorithms for the ATSP even when the costs satisfy the triangle inequality. The approximation factors of the existing algorithms for the ATSP either depend on the number of targets (18; 19; 20) or the input data (18). For example, the well known covering algorithm for the ATSP in (18) has an approximation factor of $\log(n)$ where n is the number of targets. There are also data dependent algorithms (18) with the approximation factors that depend on $\max_{i,j} \frac{c_{ij}}{c_{ji}}$ where c_{ij} denotes the cost of traveling from vertex i to vertex j.

When the travel costs are *symmetric* and satisfy the triangle inequality, authors in (21) provide an approximation algorithm for the FCURP. They assume that the minimum fuel required to travel from any target to its nearest depot is at most equal to $\frac{La}{2}$ units where a is a constant in the interval $[0, 1)$ and L is the fuel capacity of the vehicle. This is a reasonable assumption, as in any case, one cannot have a feasible tour if there is a target that cannot be visited from any of the depots. Using these assumptions, Khuller et al. (21) present a $\frac{3(1+a)}{2(1-a)}$-approximation algorithm for the problem. In this chapter, we extend this result for the asymmetric case.

FCURP is related to a more general search problem with uncertainties (22) where the fuel constraints are posed as a restriction on the time spent by the vehicle between any two successive depots on its path. The authors in (22) discretize time and space, and develop heuristics based on the shortest path algorithms. There are also variants of the vehicle routing problem that are closely related to the FCURP. For example, in (23; 24), the authors address a symmetric version of the arc routing problem where there is a single depot and a set of intermediate facilities, and the vehicle has to cover a subset of edges along which customers are present. The vehicle is required to collect goods from the customers as it traverses the given set of edges and unload the goods at the intermediate facilities. The goal of this problem is to find a tour of minimum length that starts and ends at the depot such that the vehicle visits the given subset of edges and the total amount of goods carried by the vehicle never exceeds the capacity of the vehicle at any location along the tour. One of the key differences between the arc routing problem and the FCURP is that there is no requirement that any subset of edges must be visited in the FCURP. There are also similar problems (25; 26) addressed in the literature where each customer is located at a distinct vertex (instead of being present along the edges) and the vehicle is required to collect goods from the customers and deliver them at the intermediate facilities. FCURP is also different from the single depot vehicle routing problems addressed in (27; 28; 29) where there are additional length, travel-time or capacity constraints.

In the context of the above results, we first present an algorithm for the FCURP with an approximation factor of $\frac{(1+a+2a\beta)\log(|T|)}{(1-a)}$ where T represents the set of targets, and a

and β are data dependent constants (section 3.3). We then use the solution produced by the approximation algorithm as an initial solution and apply construction/improvement heuristics (section 3.4) to obtain solutions with better quality. Computational results are then presented in section 3.5 to compare the performance of all the algorithms with respect to the quality of the solutions produced by the algorithms and their respective computation times.

3.2 Problem Statement

Let T denote the set of targets and D represent the set of depots. Let $s \in D$ be the depot where the UAV is initially located. The FCURP is formulated on the complete directed graph $G = (V, E)$ with $V = T \cup D$. Let f_{ij} represent the amount of fuel required by the vehicle to travel from vertex $i \in V$ to vertex $j \in V$. It is assumed that the fuel costs satisfy the triangle inequality *i.e.*, for all distinct $i, j, k \in V$, $f_{ij} + f_{jk} \geq f_{ik}$.

Let L denote the maximum fuel capacity of the vehicle. For any given target $i \in T$, we will assume that there are depots d_1 and d_2 such that $f_{d_1 i} + f_{i d_2} \leq aL$ where a is a fixed constant in the interval $[0, 1)$. This is a reasonable assumption, as in any case, target i cannot be visited by the vehicle if there are no depots d_1 and d_2 such that $f_{d_1 i} + f_{i d_2} > L$. We will also assume that it is always possible to travel from one depot to any another depot (either directly or by passing through some intermediate depots) without violating the fuel constraints. Given two distinct depots d_1 and d_2, let l'_{d_1,d_2} denote the minimum fuel required to travel from d_1 to d_2. Then, let β be a constant such that $l'_{d_2,d_1} \leq \beta l'_{d_1,d_2}$ for all distinct $d_1, d_2 \in D$.

A tour for the vehicle is denoted by a sequence of vertices $(s, v_1, v_2, \cdots, v_p, s)$ visited by the vehicle where $v_i \in V$ for $i = 1, \cdots, p$. Without loss of generality, we will assume that there is a target exactly at the location of the initial depot; therefore, a tour visiting all the targets can be transformed to a tour visiting all the targets and the initial depot and vice versa.

The objective of the problem is to find a $TOUR := (s, v_1, v_2, \cdots, v_p, s)$ such that

- $\{v_1, v_2, \cdots, v_p\} \supseteq T$,
- the fuel required to travel any part of the tour $(d_1, t_1, \cdots, t_k, d_2)$ $\subseteq TOUR$ starting at a depot d_1 and ending at the next visit to a depot d_2 while visiting a sequence of targets $t_1, \cdots, t_k \in T$ must be at most equal to L, *i.e.*, $f_{d_1,t_1} + \sum_{i=1}^{k-1} f_{t_i,t_{i+1}} + f_{t_k,d_2} \leq L$, and,
- the travel cost, $f_{s,v_1} + \sum_{i=1}^{p-1} f_{v_i,v_{i+1}} + f_{v_p,s}$, is a minimum.

3.3 Approximation Algorithm

We refer to this approximation algorithm as *Approx*. There are three main steps in *Approx*. The **first step** of *Approx* aims to find a path for the vehicle to travel from any target $x \in T$ to any other target $y \in T$ such that the path can be a part of a feasible

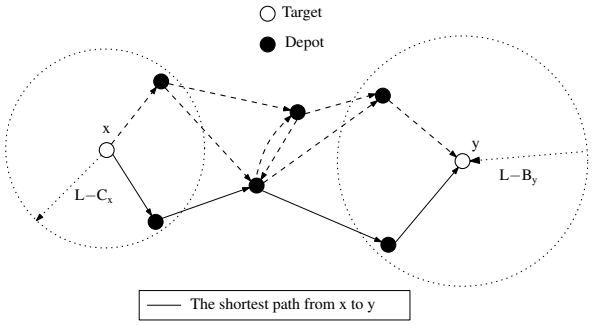


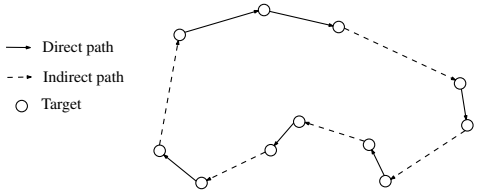
Figure 3.2: The first step of the approximation algorithm: The solid edges represent the shortest path $PATH_{xy}$ from target x to target y, and the cost of traveling this path is denoted by l_{xy}.

tour for the FCURP, the path satisfies all the refueling constraints and the travel cost associated with the path is a minimum. Note that the maximum amount of fuel available for the vehicle when it reaches target x in any tour is $L - \min_d f_{dx}$. Also, in any feasible tour, there must be at least $\min_d f_{yd}$ units of fuel left when the vehicle reaches target y so that the vehicle can continue to visit other vertices along its tour. Define $C_x := \min_d f_{dx}$ and $B_x := \min_d f_{xd}$ for any $x \in T$. The first step of the *Approx* essentially finds a feasible path of least cost (also referred as the shortest path) such that the vehicle starts at target x with at most $L - C_x$ units of fuel and ends at target y with at least B_y units of fuel. If there is enough fuel available for the vehicle to travel from x to y (or, if $L - C_x - B_y \geq f_{xy}$), the vehicle can directly reach y from x while respecting the fuel constraints. In this case, we say that the vehicle can *directly* travel from x to y and the shortest path (also referred to as the *direct* path) is denoted by $PATH(x, y) := (x, y)$. The cost of traveling this shortest path is just f_{xy}.

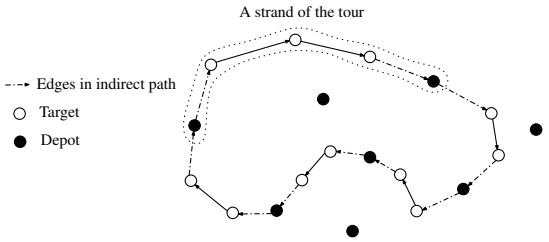
If the vehicle *cannot directly* travel from x to y (if $L - C_x - B_y < f_{xy}$), the vehicle must visit some of the depots on the way before reaching target y. In this case, we find a shortest path using an auxiliary directed graph, (V', E'), defined on all the depots and the targets x, y, *i.e.*, $V' = D \cup \{x, y\}$ (illustrated in figure 3.2). An edge is present in this directed graph only if traveling the edge can satisfy the fuel constraint. For example, as the vehicle has at most $L - C_x$ units of fuel to start with, the vehicle can reach a depot d from x only if $f_{xd} \leq L - C_x$. Therefore, E' contains an edge (x, d) if the constraint $f_{xd} \leq L - C_x$ is satisfied. Similarly, the vehicle can travel from a depot d to target y only if there are at least B_y units of fuel remaining after the vehicle reaches y. Therefore, E' contains an edge (d, y) if the constraint $f_{dy} \leq L - B_y$ is satisfied. In summary, the following are the edges present in E':

$$E' := \begin{cases} \{(x, d) : \forall d \in D, f_{xd} \leq L - C_x\}, \\ \bigcup\{(d_1, d_2) : \forall d_1, d_2 \in D, f_{d_1 d_2} \leq L\}, \\ \bigcup\{(d, y) : \forall d \in D, f_{dy} \leq L - B_y\}. \end{cases} \tag{3.1}$$

Any path starting at x and ending at y in this auxiliary graph will require the vehicle to carry at most $L - C_x$ units of fuel at target x, satisfy all the fuel constraints and reach target y with at least B_y units of fuel left. Also, we let the cost of traveling any edge



(a) A sample tour covering all the targets obtained using the covering algorithm with l_{xy} as the cost metric.



(b) The indirect edges in the tour are replaced with the corresponding shortest paths.

Figure 3.3: An illustration of the second step of the approximation algorithm.

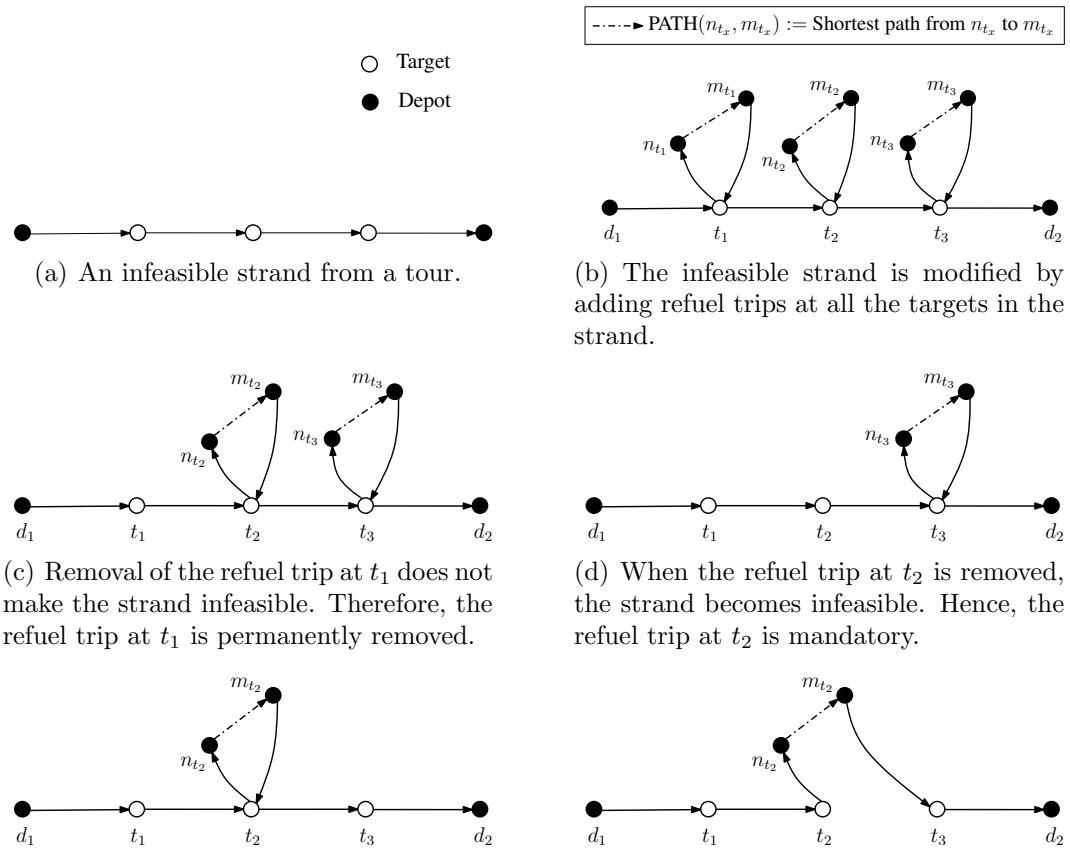
$(i, j) \in E'$ to be equal to f_{ij} (as defined in section 3.2). Now, we use Dijkstra's algorithm (30) to find a shortest path to travel from x to y. This shortest path (also referred to as the *indirect* path using intermediate depots) can be represented as $PATH(x, y) := (x, d_1, d_2, \cdots, y)$.

In the **second step** (illustrated in figure 3.3) of *Approx*, we use the shortest path computed between any two targets to find a tour for the vehicle. To do this, let l_{xy} denote the cost of the shortest path $PATH(x, y)$ that starts at x and ends at y. The following covering algorithm (18) is used to obtain a tour which visits each of the targets at least once. Suppose G'_o represent the collection of edges chosen by the covering algorithm. Initially, G'_o is an empty set.

- Let $T' := T$. Find a minimum cost cycle cover, C, for the graph (T', E'_T) with $E'_T := \{(x, y) : x, y \in T'\}$ and l_{xy} as the cost metric. A cycle cover for a graph is a collection of edges such that the indegree and the outdegree of each vertex in the graph is exactly equal to one. A minimum cost cycle cover is a cycle cover such that the sum of the cost of the edges in the cycle cover is a minimum. This step can be solved in at most $O(|T'|^3)$ steps using the Hungarian algorithm (18). Add all the edges found in C to G'_o.

- If the cycle cover consists of at least two cycles, select exactly one vertex from each cycle and return to step 1 with T' containing only the selected vertices. If the cycle cover C consists of exactly one cycle go to the next step.

- The collection of edges in G'_o represents a connected Eulerian graph spanning all the targets where the indegree and the outdegree of each target is the same. Given



(a) An infeasible strand from a tour.

(b) The infeasible strand is modified by adding refuel trips at all the targets in the strand.

(c) Removal of the refuel trip at t_1 does not make the strand infeasible. Therefore, the refuel trip at t_1 is permanently removed.

(d) When the refuel trip at t_2 is removed, the strand becomes infeasible. Hence, the refuel trip at t_2 is mandatory.

(e) Removal of the refuel trip at t_3 does not make the strand infeasible. Hence, the refuel trip at t_3 is permanently removed.

(f) The edges incident on the targets are then shortcut as the fuel costs satisfy the triangle inequality.

Figure 3.4: The greedy procedure to convert an infeasible strand to a feasible strand.

an Eulerian graph, using Euler's theorem, one can always find a tour such that each edge in G'_o is visited exactly once. This tour is the output of the covering algorithm.

If there is any edge (x, y) in this tour such that the vehicle *cannot directly* travel from x to y, (x, y) is replaced with all the edges present in the shortest path, $PATH(x, y)$, from x to y. After replacing all the relevant edges with the edges from the shortest paths, one obtains a Hamiltonian tour, $TOUR$, which visits each of the targets at least once and some of the intermediate depots for refueling. This tour may still be infeasible because there may be a sequence of vertices that starts at a depot and ends at the next depot on the tour which may not satisfy the fuel constraints. To correct this, we further augment this tour with more visits to the depots as explained in the next step of the algorithm.

In the **last step** of *Approx* (illustrated in figure 3.2), the entire tour, $TOUR$, obtained from the second step is decomposed into a series of strands. A strand is a sequence of adjacent vertices in the tour that starts at a depot, visits a set of targets and ends at a depot. $TOUR$ must be infeasible if the total fuel required to travel any one of these strands is greater than the fuel capacity of the vehicle (L). Hence, in this step, all the infeasible strands are identified, and a greedy algorithm is applied to each infeasible strand

to transform it to a feasible strand (refer to figure 3.4). We present some definitions before we outline the greedy algorithm. A depot, m_x, is referred as a *nearest starting* depot for x if $f_{m_x x} = \min_d f_{dx}$. Similarly, a depot n_x is referred as a *nearest terminal depot* for x if $f_{x n_x} = \min_d f_{xd}$. As in the second step of the algorithm, given any two depots $d_s, d_f \in D$, one can find a path of least cost that starts from d_s, visits some intermediate depots (if necessary) and ends at d_f while satisfying all the fuel constraints [1]. Let the sequence of all the depots in this path be denoted by $PATH(d_s, d_f) := (d_s, \overline{d}_1, \overline{d}_2, \cdots, \overline{d}_k, d_f)$ where $\overline{d}_1, \overline{d}_2, \cdots, \overline{d}_k \in D$ are the intermediate depots visited by the vehicle.

The greedy algorithm works as follows (refer to figure 3.4): Consider an infeasible strand represented as $(d_1, t_1, \cdots, t_k, d_2)$ where d_1 and d_2 are the two depots of the strand and $t_1, \cdots, t_k$ are the targets. For each target t in this infeasible strand, we add a refueling trip such that

- The vehicle visits a nearest terminal depot n_t after leaving t.
- The vehicle uses the sequence of depots specified in $PATH(n_t, m_t)$ to travel from n_t to m_t where m_t is the nearest starting depot for t, and finally returns to t after refueling.

After adding all the refueling trips, the modified strand can be denoted as $(d_1, t_1, PATH(n_{t_1}, m_{t_1}), t_1, t_2, PA$ $t_2, \ldots, PATH(n_{t_k}, m_{t_k}), t_k, d_2)$. Now, each of the refueling trips is chosen sequentially in the order they are added and is shortcut if the strand that results after removing the refueling trip still satisfies the fuel constraint (refer to figure 3.4).

3.3.1 Analysis of the Approximation Algorithm

Lemma 3.3.1 *Approx always produces a feasible solution for the FCURP.*

Proof Consider the greedy procedure presented in the last step of the *Approx* which attempts to convert an infeasible strand $(d_1, t_1, t_2, \cdots, t_k, d_2)$ into a feasible path for the vehicle. The edges (d_1, t_1) and (t_k, d_2) in this strand belong to indirect paths while the remaining edges belong to direct paths. The vehicle can always travel from d_1 to t_1 and still have enough fuel at t_1 to reach its nearest terminal depot as edge (d_1, t_1) was added according to the fuel constraints in (3.1). Therefore, once the vehicle reaches t_1, due to our assumptions on the fuel costs, there always exists a refueling trip such that the vehicle starts at t_1, visits the depots n_{t_1}, m_{t_1} before returning to t_1 with the maximum amount of fuel possible at t_1. As a result, the vehicle must be able to reach t_2 with sufficient amount of fuel remaining to reach the nearest terminal depot n_{t_2}. Again, there exists a refueling trip such that at the end of this trip, the vehicle can return to t_2 with the maximum amount of fuel possible at t_2. The above arguments can be repeatedly used for each target in the infeasible strand to show that the vehicle must be able to reach d_2 using the modified strand while satisfying the fuel constraints. Therefore, the greedy procedure can always convert any infeasible strand into a feasible path and hence, the *Approx* finds a feasible solution to the FCURP.

[1] Apply Dijkstra's algorithm on the graph (D, E_d) where $E := \{(i, j) : i, j \in D, f_{ij} \leq L\}$ and the cost of traveling from vertex $i \in D$ to vertex $j \in D$ is c_{ij}.

The cost of the final solution (say $TOUR_f$) obtained by *Approx* is upper bounded by the sum of the cost of $TOUR$ and the cost of all the refueling trips. So, in order to bound the cost of $TOUR_f$, we need to bound the cost of $TOUR$, the number of refueling trips and the cost of each refueling trip in terms of the optimal cost of the FCURP. In the following lemma, we first bound the cost of $TOUR$.

Lemma 3.3.2 *Let cost($TOUR$) denote the total fuel required to travel all the edges in $TOUR$. Then, cost($TOUR$) is at most equal to* $\log(|T|) \times C_{opt}$ *where C_{opt} is the optimal cost of the FCURP.*

Proof The cost of $TOUR$ is equal to the sum of the cost of all the cycle covers spanning all the targets with l_{xy} as the cost metric. Now, consider any minimum cost cycle cover C spanning the targets $t_1, t_2, \cdots, t_m$. Without loss of generality, we also let $(t_1, t_2, \cdots, t_m)$ denote the sequence in which the targets in C are visited in an optimal solution to the FCURP. The minimum cost, $l_{t_i,t_{i+1}}$, of traveling from target t_i to t_{i+1} (computed in the first and the second step of *Approx*) must be at most equal to the cost of traveling from t_i to t_{i+1} in the optimal solution of the FCURP. Therefore, the minimum cost of a TSP tour visiting any subset of targets using l_{xy} as the metric must be at most equal to C_{opt}. Since the problem of computing a minimum cost cycle cover is a relaxation to the TSP, it follows that the cost of any optimal cycle cover computed in the second step of *Approx* must be at most equal to C_{opt}. The number of iterations in the covering algorithm is at most $\log(|T|)$ as the number of selected targets in any two successive iterations of the covering algorithm reduces by half. Hence, the cost of $TOUR$ which is the same as the total cost of all the cycle covers is at most equal to $\log(|T|) \times C_{opt}$.

In the following lemma, we bound the number of refueling trips needed to make $TOUR$ feasible.

Lemma 3.3.3 *The number of refueling trips needed by the vehicle is upper bounded by* $\frac{2cost(TOUR)}{(1-a)L}$.

Proof Let $I = (d_1, t_1, t_2, \cdots, d_2)$ represent an infeasible strand in $TOUR$ that requires additional refueling trips and let $cost(I)$ denote the total fuel required to travel the edges connecting any two adjacent vertices in I. Given any two vertices $u, v \in I$ and the segment I_{uv} of I starting at u and ending at v, let $cost(u, v)$ denote the total fuel required to travel the edges connecting any two adjacent vertices in I_{uv}. Let the greedy procedure add refueling trips at targets $v_1, v_2, \cdots, v_k$ to make I feasible. Then, $cost(d_1, v_2)$ must be greater than $L - B_{v_2}$ (recall that for any target x, $C_x := \min_d f_{dx}$ and $B_x := \min_d f_{xd}$); if this is not the case, the refueling trip at target v_1 is unnecessary and can be removed. Similarly, $\text{cost}(v_1,v_3)$ must be greater than $L - C_{v_1} - B_{v_3}$, else, the refueling trip at v_2 is avoidable and can be removed. Repeating the above arguments for the pairs of vertices

$(v_2, v_4), \cdots, (v_{k-2}, v_k)$ and (v_{k-1}, d_2), we get, the following inequalities:

$$\begin{aligned}
cost(d_1, v_2) &> L - B_{v_2}, \\
cost(v_1, v_3) &> L - C_{v_1} - B_{v_3}, \\
cost(v_2, v_4) &> L - C_{v_2} - B_{v_4}, \\
\vdots \quad &\vdots \quad \vdots \\
cost(v_{k-2}, v_k) &> L - C_{k-2} - B_{v_k}, \\
cost(v_{k-1}, d_2) &> L - C_{k-1}.
\end{aligned}$$

Now, the number of refueling trips can be bounded in the following way:

$$\begin{aligned}
2cost(I) &\geq cost(d_1, v_2) + \sum_{i=1}^{k-2} cost(v_i, v_{i+2}) + cost(v_{k-1}, d_2) \\
&\geq kL - C_{v_1} - \sum_{x=v_2,\cdots,v_{k-1}} (B_x + C_x) - B_{v_k}. \qquad (3.2)
\end{aligned}$$

For any target x, as there are depots $\widehat{d}$ and $\overline{d}$ such that $f_{\widehat{d}x} + f_{x\overline{d}} \leq aL$, we have $C_x + B_x = \min_d f_{dx} + \min_d f_{xd} \leq f_{\widehat{d}x} + f_{x\overline{d}} \leq aL$. Using this bound for each target in (3.2), we get

$$\begin{aligned}
2cost(I) &\geq kL - aL - (k-2)aL - aL \qquad (3.3) \\
&\geq k(1-a)L. \qquad (3.4)
\end{aligned}$$

As a result, the number of refueling trips for strand I is upper bounded by $\frac{2cost(I)}{(1-a)L}$. Therefore, the total number of refueling trips for the infeasible strands is upper bounded by $\frac{2}{(1-a)L} \sum_I cost(I) \leq \frac{2}{(1-a)L} cost(TOUR)$.

The following theorem provides an approximation factor for *Approx* which depends on the size of the problem and the input data.

Theorem 3.3.1 *Approx solves the FCURP with an approximation factor of* $\frac{(1+a+a\beta)\log(|T|)}{1-a}L$ *in* $O(|D|^2|T|^2 + |T|^3 log(|T|))$ *steps.*

Proof The cost of the solution, $TOUR_f$, obtained by *Approx* is upper bounded by the sum of the cost of $TOUR$ and the cost of all the refueling trips. Note that the cost of the refueling trip at any target x must be equal to $f_{x\widehat{d}_1} + f_{\widehat{d}_1\widehat{d}_2} + f_{\widehat{d}_2 x}$ where the depots $\widehat{d}_1, \widehat{d}_2$ are such that $f_{x\widehat{d}_1} = \min_d f_{xd}$, $f_{\widehat{d}_2 x} = \min_d f_{dx}$. From the assumptions in section 3.2, we get,

$$\begin{aligned}
f_{x\widehat{d}_1} + f_{\widehat{d}_1\widehat{d}_2} + f_{\widehat{d}_2 x} &\leq f_{x\widehat{d}_1} + \beta f_{\widehat{d}_2\widehat{d}_1} + f_{\widehat{d}_2 x} \\
&\leq f_{x\widehat{d}_1} + \beta(f_{\widehat{d}_2 x} + f_{x\widehat{d}_1}) + f_{\widehat{d}_2 x} \\
&= (1+\beta)(f_{\widehat{d}_2 x} + f_{x\widehat{d}_1}) \\
&\leq (1+\beta)aL.
\end{aligned}$$

Using lemma 3.3.3, we can conclude that the total cost of all the refueling trips must be at most equal to $(1+\beta)aL \times \frac{2cost(TOUR)}{(1-a)L} = \frac{2(1+\beta)a}{(1-a)} cost(TOUR)$. Therefore, the total cost of

$TOUR_f$ is upper bounded by $cost(TOUR)+\frac{2(1+\beta)a}{(1-a)}cost(TOUR) = \frac{(1+a+2\beta a)}{(1-a)}cost(TOUR)$. Using lemma 3.3.2, we get, $cost(TOUR_f) \leq \frac{(1+a+2\beta a)}{(1-a)}\log(|T|)C_{opt}$. Also, the number of steps involved in the algorithm is dominated by the first and second step of *Approx*. For any given pair of targets x and y, the Dijkstra's algorithm requires at most $O(|D|^2)$ steps to compute l_{xy}. As a result, the total number of steps required to implement the first step of *Approx* is $O(|D|^2|T|^2)$. The second step of *Approx* runs the Hungarian algorithm for at most $log(|T|)$ iterations. Hence, the number of steps required to implement the second step is $O(|T|^3 log(|T|))$. Therefore, the total number of steps involved in *Approx* is $O(|D|^2|T|^2 + |T|^3 log(|T|))$.

3.4 Construction and Improvement Heuristics

The construction heuristic we propose is exactly the same as *Approx* except for its second step. Specifically, we replace the covering algorithm in the second step of *Approx* with the Lin-Kernighan-Helgaun (LKH) heuristic (31). We then use the solution obtained using the construction heuristic as an initial feasible solution for the improvement heuristics. The improvement heuristics relies on a combination of a $k-$opt heuristic and a depot exchange heuristic to improve the quality of the tour obtained by the construction heuristic. A $k-$opt heuristic is a local search method which iteratively attempts to improve the quality of a solution until some termination criteria are met. The depot exchange heuristic aims to replace some depots in the tour with refueling depots not present in the tour in order to obtain better feasible solutions. A flow chart of the overall procedure is

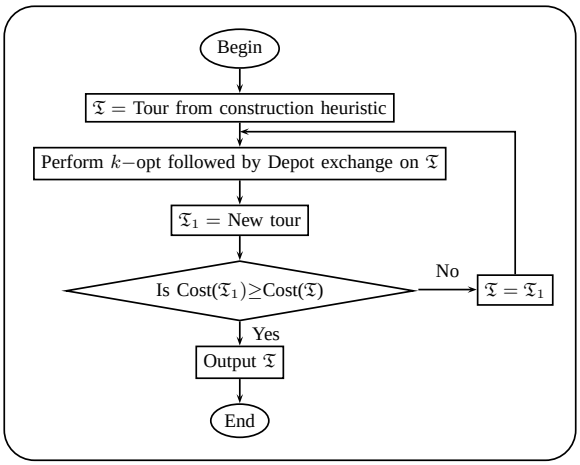
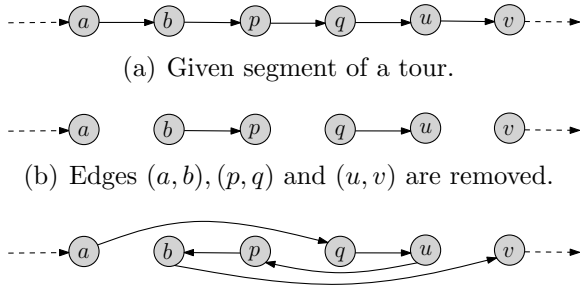


Figure 3.5: Overall procedure in the improvement heuristic.

presented in figure 3.5. In the following subsections, we explain the $k-$opt and the depot exchange heuristic in detail.

3.4.1 $k-$opt

We will first give some basic definitions involved in a $k-$opt heuristic, and then see how it is applicable to the FCURP. A tour S_2 is defined to be in the $k-$exchange neighborhood



(a) Given segment of a tour.

(b) Edges $(a, b), (p, q)$ and (u, v) are removed.

(c) One possible way of adding new edges to construct a new segment.

Figure 3.6: Possible 3-exchange move.

Algorithm 1 : Pseudo code for the $k-$opt algorithm

Notations: Let $cost(\mathfrak{T})$ denote the sum of the cost of traveling all the edges in the tour $\mathfrak{T}$. Let n denote the *search span* of a segment.

$\mathfrak{T}^* \leftarrow$ Initial feasible tour.
$\mathfrak{T} \leftarrow \mathfrak{T}^*$.
loop
  $N_d \leftarrow$ Number of visits to the depots in $\mathfrak{T}$.
  for $i = 1, \cdots, N_d$ **do**
    $\mathfrak{S}(i, n) \leftarrow$ segment of $\mathfrak{T}$ centered at the i^{th} depot visited in $\mathfrak{T}$.
    Find a tour $\mathfrak{R}$ such that for $2 \leq k' \leq k$,
      $\mathfrak{R}$ is obtained by replacing k' edges in the segment $\mathfrak{S}(i, n)$ with k' new edges;
      $\mathfrak{R}$ is the best improving $k'-$exchange of $\mathfrak{T}$.
    If $cost(\mathfrak{R}) < cost(\mathfrak{T})$, $\mathfrak{T} \leftarrow \mathfrak{R}$.
  end for
  if $cost(\mathfrak{T}^*) \leq cost(\mathfrak{T})$ **then**
    break;
  else
    $\mathfrak{T}^* \leftarrow \mathfrak{T}$.
  end if
end loop
Output $\mathfrak{T}^*$ as the solution.

of the tour S_1 if S_2 can be obtained from S_1 by replacing k edges in S_1 with k new edges. A tour S_2 is said to be obtained from a feasible tour S_1 by an improving $k'-$exchange if S_2 is in the $k'-$exchange neighborhood of S_1, is feasible and has a travel cost lower than S_1. The $k-$opt heuristic starts with a feasible tour and iteratively improves on this tour making successive improving $k'-$exchanges for any $2 \leq k' \leq k$ until no such exchanges can be made.

A critical part of developing a $k-$opt heuristic deals with choosing an appropriate $k'-$exchange neighborhood for a tour. One way to choose this is to consider all possible subsets of k' edges in the tour and try an improving $k'-$exchange. Initial implementations showed us that substantial improvements in the quality of the tour were obtained when the $k'-$exchanges where performed around the refueling depots in the tour. In view of this observation, we define a segment of span n as a sequence of $2n+1$ adjacent vertices of the tour centered around a depot. A segment can be denoted by $(s_1, \ldots, s_n, d, s_{n+1}, \ldots, s_{2n})$, where d is the depot around which the segment is centered. Following the definition of a segment, one can infer that the number of possible segments in a feasible tour is equal to the number of visits by the UAV to all the depots.

The $k'-$exchange neighborhood in each iteration is restricted to one of the segments of the given tour. Given a segment, k' edges are deleted from the segment, and subsequently k' new edges are added to form a new segment as shown in figure 3.6. The updated tour is then checked for feasibility to ensure that the UAV never runs out of fuel. The pseudo code for the $k-$opt heuristic is shown in algorithm 1.

3.4.2 Depot Exchange Heuristic

Given a tour, we consider the depots in the order in which they are visited by the UAV and substitute each of them with a (possibly) new refueling depot in order to obtain a better feasible solution. For a given depot d in the tour, suppose v_1 and v_2 are the vertices that are visited immediately before and after visiting d in the tour. The heuristic replaces d with a new depot $d_r := \underset{u \in D}{\operatorname{argmin}} \; c_{v_1 u} + c_{u v_2}$ if the new tour is feasible and reduces the total cost. The new tour then acts as the current feasible solution and the above procedure is repeated for each depot until no further improvements can be done.

3.5 Computational Results

We considered problems of size ranging from 10 targets to 25 targets with increments in steps of 5. For each problem size, 50 instances were generated and all the targets were chosen randomly from a square area of 5000×5000 units. In addition, all the instances of the problem have 5 depots chosen at fixed locations in the square area. All the simulations were run on a Dell Precision T5500 workstation (Intel Xeon E5630 processor @ 2.53GHz, 12GB RAM).

The simulations were performed for a fixed wing vehicle with minimum turning radius constraints. A vehicle traveling at a constant speed with a bound on its turning radius is referred to as the Dubins' vehicle (14). In the simulations, the minimum turning radius of the vehicle is chosen to be 100 units and the angle of approach for each target is selected

uniformly in the interval $[0, 2\pi]$ radians. Given the approach angles at the targets, the minimum distance required to travel between any two targets subject to the turning radius constraints of the vehicle was solved by Dubins in (14). For the simulations, the maximum fuel capacity L was 4500 units and we assumed that the fuel spent is directly proportional to the distance traveled by the vehicle.

To find the optimal solution to the FCURP, the formulation presented in the appendix was solved to optimality using the IBM ILOG CPLEX optimization software (32). The approximation algorithm and the heuristics were coded using Python 2.7.2 (33). We used a search span of 4 for the improvement heuristics as it gave a good trade off between the solution quality and the computation time available. The average time required to find an optimal solution in CPLEX was nearly 2 hours for problem instances with 25 targets and 5 depots. On the other hand, the average time required to find a feasible solution using the approximation algorithm and the heuristics was less than 2 seconds for each tested instance.

As mentioned in the introduction, the approximation factor of *Approx* provides an a-priori, theoretical upper bound on the quality of the solutions obtained by *Approx*. For the tested instances, the approximation factor of *Approx* was 83.18, 87.06, 131.79 and 180.68 for 10, 15, 20 and 25 targets respectively. On the other hand, the *worst-case* ratio of the cost of the solution produced by the approximation algorithm to the optimal cost was 1.52, 1.61, 1.82 and 1.74 for 10, 15, 20 and 25 targets respectively. These results imply that *Approx* finds solutions with bounds that are significantly better than the guarantees indicated by the approximation factor.

In addition to the approximation factors, we also computed the average deviation of the sub-optimality of the feasible solutions produced by the algorithms using the following metric: The deviation in the cost of the solution produced by applying an algorithm on an instance I is defined as $100 \times \frac{C_I^{algorithm} - C_I^{optimal}}{C_I^{optimal}}$ where $C_I^{algorithm}$ is the cost of the solution found by the algorithm and $C_I^{optimal}$ is the cost of the optimal solution for an instance I.

The average deviation of the solutions produced by the approximation algorithm and the heuristics for the instances is shown in the figure 3.7. From the figure, it is clear that the average deviation of the solutions produced by the improvement heuristic is much superior compared to the average deviation of the solutions found by the construction heuristic or the approximation algorithm. The depot-exchanges played a substantial part in improving the quality of the solutions found by the improvement heuristics; in particular, on an average, the depot exchange reduced the deviation by 0.14%, 0.66%, 0.78% and 1.10% for problems with 10, 15, 20 and 25 targets respectively. The feasible solution produced by the improvement heuristic was also used as an initial feasible solution for the formulation in CPLEX. The formulation was then solved in CPLEX with a time bound of 10 seconds. Using the feasible solution produced by the heuristic as a starting point, CPLEX was able to further reduce the deviation of the solutions as shown in figure 3.7. Specifically, for instances with 25 targets and 5 depots, CPLEX was further able to reduce the average deviation to 1.39%. These computational results show that the proposed algorithms can be effectively used in conjunction with standard optimization software like CPLEX in order to obtain high quality solutions for the FCURP.

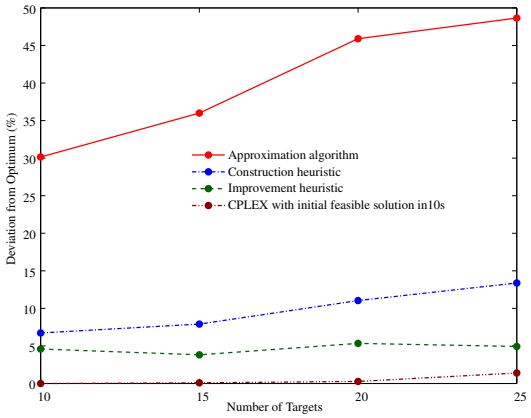


Figure 3.7: Average deviation of the solutions produced by the proposed algorithms.

3.6 Conclusions

An approximation algorithm and fast heuristics were developed to solve a generalization of the single vehicle routing problem with fuel constraints. A mixed-integer, linear programming formulation was also proposed to find optimal solutions to the problem. Future work can be directed towards developing branch and cut methods, and can address problems with multiple, heterogeneous vehicles.

3.7 Appendix

Let x_{ij} denote an integer decision variable which determines the number of directed edges from vertex i to j in the network; that is, x_{ij} is equal to q if and only if the vehicle travels q times from vertex i to vertex j. As the costs satisfy the triangle inequality, without loss of generality, we can assume that there is an optimal solution such that each target is visited exactly once by the vehicle. Therefore, we restrict $x_{ij} \in \{0, 1\}$ if either vertex i or vertex j is a target.

The collection of edges chosen by the formulation must reflect the fact that there must be a path from the depot to every target. We use flow constraints (34) to formulate this connectivity constraint. In these flow constraints, the vehicle collects $|T|$ units of a commodity at the depot and delivers one unit of commodity at each target as it travels along its path. Enforcing that these commodities can be routed through the chosen edges ensures there is a path from the depot to every target. Suppose p_{ij} denotes the amount of commodity flowing from vertex i to vertex j. Also, let r_i represent the fuel left in the vehicle when the i^{th} target is visited. The FCURP can be formulated as a mixed integer linear program as follows:

$$\min \sum_{(i,j)\in E} c_{ij}\ x_{ij}$$

subject to *Degree constraints:*

$$\sum_{i \in V \setminus \{k\}} x_{ik} = \sum_{i \in V \setminus \{k\}} x_{ki} \quad \forall k \in V, \tag{3.5}$$

$$\sum_{i \in V \setminus \{k\}} x_{ik} = 1 \quad \forall k \in T, \tag{3.6}$$

Capacity and flow constraints:

$$\sum_{i \in V \setminus \{s\}} (p_{si} - p_{is}) = |T|, \tag{3.7}$$

$$\sum_{j \in V \setminus \{i\}} (p_{ji} - p_{ij}) = 1 \quad \forall i \in T, \tag{3.8}$$

$$\sum_{j \in V \setminus \{i\}} (p_{ji} - p_{ij}) = 0 \quad \forall i \in D \setminus \{s\}, \tag{3.9}$$

$$0 \leq \ p_{ij} \ \leq |T| x_{ij} \quad \forall i, j \in V, \tag{3.10}$$

Fuel constraints:

$$r_j - r_i + f_{ij} \leq M(1 - x_{ij}) \quad \forall i, j \in T, \tag{3.11}$$

$$r_j - r_i + f_{ij} \geq -M(1 - x_{ij}) \quad \forall i, j \in T, \tag{3.12}$$

$$r_j - L + f_{ij} \geq -M(1 - x_{ij}) \quad \forall i \in D \text{ and } j \in T, \tag{3.13}$$

$$r_j - L + f_{ij} \leq M(1 - x_{ij}) \quad \forall i \in D \text{ and } j \in T, \tag{3.14}$$

$$r_i - f_{ij} \geq -M(1 - x_{ij}) \quad \forall i \in T \text{ and } j \in D, \tag{3.15}$$

$$\begin{aligned} &0 \leq r_i \leq L \quad \forall i \in T, \\ &x_{ij} \in \{0, 1\} \quad \forall \ i, j \in V, \text{ either } i \text{ or } j \text{ is a target}, \\ &x_{ij} \in \{0, 1, 2, \cdots, |T|\} \quad \forall \ i, j \in D. \end{aligned} \tag{3.16}$$

Equation (3.5) states that the in-degree and out-degree of each vertex must be the same, and equation (3.6) ensures that each target is visited once by the vehicle. Note that these equations allow for the vehicle to visit a depot any number of times for refueling. The constraints in (3.7)-(3.10) ensure that there are $|T|$ units of commodity shipped from the depot and the vehicle delivers exactly one of commodity at each target. In equations, (3.11)-(3.15), M denotes a large constant and can be chosen to be equal to $L + \max_{i,j \in V} f_{i,j}$. If the UAV is traveling from target i to target j, equations (3.11) and (3.12) ensure that the fuel left in the vehicle after reaching target j is $r_j = r_i - f_{ij}$. If the UAV is traveling from depot i to target j, equations (3.13), (3.14) ensure that the fuel left in the vehicle after reaching target j is $r_j = L - f_{ij}$. If the UAV is directly traveling from any target to a depot, constraint (3.15) states that the fuel remaining at the target must be at least equal to the amount required to reach the depot.

Chapter 4

Stochastic Dynamic Programming

4.1 Introduction

This chapter is motivated by a robotic perimeter surveillance problem. A collection of robots assists a remotely located human operator in the task of classification of an incursion, across the perimeter of a protected zone, as either a nuisance or a threat. Incursions are stochastic and have both a spatial and temporal component; we assume that the statistics of the incursion processes are known.

In order to aid the robot-operator team in the timely classification of incursions, the perimeter is installed with a set of Unattended Ground Sensors (UGSs) at locations where incursions can occur; these locations will be referred to as stations or UGS stations. At the stations, UGS flag incursions, raise alerts and communicate them immediately to the robots. Subsequently, a robot services the alert by visiting the UGS stations where it was raised, and transmitting images, video, or other sensory information to the operator using on-board camera and other sensing devices. The operator performs the role of a classifier based on the information supplied by the robots. The classification accuracy of the operator depends on the volume and freshness of information supplied by the robots.

For an accurate classification, the robot should provide as much video or other evidence about the incursion to the operator as possible. Subject to certain limits, the longer a robot spends at an alert location, the robot supplies a higher volume of information about the alert it services. However, the freshness of information it can gather about other unserviced alerts suffers. For timely and accurate classification of incursions, the *delay time*, defined as the time delay between an alert signal and the time a robot attends to the alert, should be minimized. Thus, there are two competing needs: a robot needs to spend more time at an alert location and it also needs to service the alerts as quickly as possible. A natural question arises: How long should a robot spend time servicing an alert?

In this chapter, we discretize the problem spatially and temporally and recast the optimization problem as follows: Should the robot spend the next time interval at the alert location in terms of maximizing the expected, discounted payoff? The payoff considered herein is an increasing function of the time spent at the alert site (dwell time) and a decreasing function of the delay in servicing alerts.

This problem be naturally posed as a Markov Decision Problem (MDP). However, the number of states runs into billions even for a modest size problem. For example, if one considers two robots and eight alert locations, with a maximum allowable delay time of 30 units, the number of states exceeds 30 billion! Hence, solving Bellman's equation to compute the optimal payoff (value function) is computationally intractable. For this reason, we consider a Linear Programming (LP) based approximate dynamic programming solution strategy (see (35)). This approach provides an upper bound on the optimal value function, and an estimate of the quality of the resulting sub-optimal policy, e. g., see (36; 37).

4.1.1 Relationship to existing Literature

Perimeter surveillance problems arise in a variety of practical applications and have recently received significant attention in the literature; for example, see (38; 39; 40; 41). The results described in this chapter and our prior work in (42; 43; 44; 45; 46) differ from the literature in addressing the need to balance the information gained by the robots with the quality of service requirement of attending to alerts raised at the UGS locations in a timely manner.

This chapter builds on the conference version of our paper ((47)) and differs from it in three ways: From an analytical point of view, it provides a performance guarantee of the sub-optimal policy. From the point of view of application, we consider the case of two robots in surveillance instead of a single robot. From the organization point of view, we have tried to provide a distinction between properties that hold for general MDPs and those that exploit the structure of the robotic surveillance/patrol problem considered in this chapter.

In terms of our previous work, only one of our papers, (43), deals with multiple (two) robots; it focuses on the computation of the optimal policy, while the focus of the present chapter is to develop sub-optimal policies with bounds. The computational complexity of determining the optimal policy does not scale well with the size of the perimeter patrol problem and hence, one must consider sub-optimal policies even for problems of modest size.

The use of LP techniques for solving Dynamic Programming (DP) problems was introduced by (48; 49); the use of aggregation via partitioning and the construction of sub-optimal policies using approximate value functions was discussed in (50). The LP based approach to approximate dynamic programming is discussed in (35; 36; 37). The results in this chapter differ from the existing literature on two counts: (1) the restricted or constrained LPs that one obtains for this class of applications are computationally tractable and hence, there is no need for column generation or random sampling techniques as in Trick and Zin (1993), and (2) this work presents a way to construct upper as well as lower bounds using LPs, a marked departure from prior work in this area, other than by the authors' work in (51). Also, the results in this chapter differ from the earlier work of the authors given in (46); in this work, we provide a performance guarantee on the sub-optimal policy constructed via the lower bound.

The chapter is organized as follows: In Section 4.2, we present the mathematical

formulation for the class of problems considered. In Section 4.4, we present the main results on the upper and lower bounds and the performance guarantees in the form of theorems. In section 4.5, we present an illustrative example of a perimeter patrol problem and showcase the results.

4.2 Problem Formulation

We will discretize the problem spatially and temporally; nodes on the perimeter partition the perimeter uniformly. The distance between adjacent nodes on the perimeter is a unit of length and the time taken by a robot to traverse between two adjacent nodes is the unit of time. The UGS locations form a subset of the nodes.

Let $\mathbf{x}_r(t), \mathbf{u}(t)$ denote the states of n_r robots in the collection and their control actions respectively at time t. Let $\mathbf{x}_s(t), \mathbf{d}(t)$ denote the states associated with the n_s UGS locations and the disturbance (incursions) occuring at those locations respectively. One may associate the states of the robots with their location, direction of travel around the perimeter, the amount of time they spend (or dwell) servicing an alert at the UGS location etc. One may think of the states associated with UGS locations to be the delays incurred in servicing the alerts raised at those locations. The control actions of the robots at time t are captured by the vector $\mathbf{u}(t)$; a sample control action indicates whether a robot should dwell at its current location or continue in the same direction or reverse. The disturbance $\mathbf{d}(t)$ can take any of the possible L values, namely $\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_L$ with corresponding probabilities $p_1, \ldots, p_L$; these probabilities are assumed to be known a priori. The number of possible values the disturbance $\mathbf{d}(t)$ can take depends on the model of incursion processes; for example, if at most one incursion is allowed at any time across the n_s stations, then $L = n_s + 1$; if, on the other hand, incursions can occur simultaneously at all the stations, then $L = 2^{n_s}$. The latter model of incursion allows one to relate this problem of perimeter surveillance to the cyclic poll server model, for example, see (52); however, their work corresponds to the robot serving alerts in a cyclical manner without having to take any decisions to stay put at the location or to reverse.

For some suitable vector fields, f_r and f_s, one may write the equations governing the evolution of the states $\mathbf{x}_r$ and $\mathbf{x}_s$ as:

$$\mathbf{x}_r(t+1) = f_r(\mathbf{x}_r(t), \mathbf{u}(t)), \tag{4.1}$$

$$\mathbf{x}_s(t+1) = f_s(\mathbf{x}_r(t), \mathbf{x}_s(t), \mathbf{u}(t), \mathbf{d}(t)). \tag{4.2}$$

For the sake of notational convenience, let the state of the system $\mathbf{x}(t) := (\mathbf{x}_r(t), \mathbf{x}_s(t))$. We may combine the evolution equations (4.1) and (4.2) as:

$$\mathbf{x}(t+1) = f(\mathbf{x}(t), \mathbf{u}(t), \mathbf{d}(t)), \tag{4.3}$$

for some appropriate vector field f.

Additionally, there may be constraints on the state and control input, of the form:

$$\mathbf{g}(\mathbf{x}(t), \mathbf{u}(t)) \leq 0, \quad \forall t \geq 0, \tag{4.4}$$

which model the allowable control actions that can be taken by the robots. For example, the state of the stationary UGS sensors can only be altered by the actions/inputs of a robot that has spent a pre-specified amount of time in its neighborhood.

Let $r(\mathbf{x}, \mathbf{u})$ denote the one-step payoff/ reward associated with the state $\mathbf{x}$ and the control input $\mathbf{u}$. We focus our attention on stationary feedback policies, $\pi \in \Pi := \mathcal{S} \times \mathcal{U}$, i.e., $\mathbf{u} = \mathbf{u}_\pi(\mathbf{x})$. Now consider the stochastic optimization problem: For a specified discount factor, $\lambda \in (0, 1)$, find a stationary policy, π, such that the following objective is maximized:

$$V^*(\mathbf{x}^0) := \max_{\pi \in \Pi} \mathbf{E}[\sum_{t=0}^{\infty} \lambda^t r(\mathbf{x}(t), \mathbf{u}_\pi(\mathbf{x}(t)))|\mathbf{x}(0) = \mathbf{x}^0], \tag{4.5}$$

where Π is the set of all possible stationary policies.

The value function $\mathbf{V}^*$ is a vector with a component $V^*(\mathbf{x}^0)$ corresponding to the initial state $\mathbf{x}^0$. It is well-known that $\mathbf{V}^*$ satisfies Bellman's equation; however, the computational tractability depends on the dimension of $\mathbf{V}^*$ given by $|\mathcal{S}|$. For a modest size problem involving 2 robots and 8 stations, the value of $|\mathcal{S}|$ can be upwards of 20 billion. For this reason, the conventional techniques of value and policy iterations to solving Bellman's equation are unsuitable.

In order to distinguish between properties that are valid for general MDPs and the MDP for perimeter patrol application, we will let $p(\mathbf{s}, \mathbf{u}, \mathbf{z})$ represent the probability of state $\mathbf{s}$ transitioning to $\mathbf{z}$ under the influence of $\mathbf{u}$ for a general MDP; for the current application, if for some l, we have $\mathbf{z} = \mathbf{f}(\mathbf{s}, \mathbf{u}, \mathbf{y}_l)$, then $p(\mathbf{s}, \mathbf{u}, \mathbf{z}) = p_l$; otherwise, it is zero.

4.2.1 Structure of the problem used in Approximate Dynamic Programming

We make the following additional assumptions about the structure of the system:

- **Assumption 1:** The set of allowed control actions for each robot are identical and finite, and is represented by $\mathcal{U}_r$. The vector $\mathbf{u}$ may be expressed as $\mathbf{u} = (u_1, u_2, \ldots, u_{n_r}) \in \mathcal{U}_r^{n_r} =: \mathcal{U}$.

- **Assumption 2:** Since the problem has been discretized, let $\mathcal{S}_r$ and $\mathcal{S}_s$ represent sets of all possible *discrete* states of robots and stations respectively; since the perimeter is compact and since the disturbances and control decisions are finite, we will assume that the sets $\mathcal{S}_r$ and $\mathcal{S}_s$ are also finite. The state space $\mathcal{S}$ of the system is the Cartesian product $\mathcal{S}_r \times \mathcal{S}_s$ and is also finite.

- **Assumption 3:** In the perimeter surveillance application, we treat the delay in servicing an alert at a location as the state $\mathbf{x}_s$. In this case, the delay increases monotonically until it is reset by the action of the robots. For a given $\mathbf{x}_r, \mathbf{d}, \mathbf{u}$, the function f_s is monotone in $\mathbf{x}_s$, i.e., if $\mathbf{x}_s \geq \mathbf{z}_s$ then $f_s(\mathbf{x}_r, \mathbf{x}_s, \mathbf{u}, \mathbf{d}) \geq f_s(\mathbf{x}_r, \mathbf{z}_s, \mathbf{u}, \mathbf{d})$.

- **Assumption 4:** We assume the following structure for the one-step payoff function:

$$r(\mathbf{x}, \mathbf{u}) = \psi_r(\mathbf{x}_r, \mathbf{u}) - \psi_s(\mathbf{x}_s), \tag{4.6}$$

  where $\psi_r : \mathcal{S}_r \times \mathcal{U} \to \Re$, and $\psi_s : \mathcal{S}_s \to \Re_+$. The function ψ_r is a monotone function of $\mathbf{x}_r$ for every $\mathbf{u}$, while the function ψ_s is a monotone function of $\mathbf{x}_s$.

  This structure is motivated by the following consideration: the information gained by robots depends on how long they dwell at a station, while there is a penalty associated with tardiness in servicing alerts at other locations.

- **Assumption 5:** Each robot knows the complete state, $\mathbf{x}_s$, of all stations. While this may not be realistic, we make this assumption in order to avoid complexities that arise from incomplete information.

The structure of the perimeter patrol problem that we will exploit specifically are:

- **Evolution Invariance Property:** Assumption 3 suggests the following partial ordering of states:

$$\mathbf{x} \geq \mathbf{z} \iff \mathbf{x}_r = \mathbf{z}_r, \;\; \mathbf{x}_s \geq \mathbf{z}_s.$$

  Assumption 3 implies that if two initial states, $\mathbf{x}, \mathbf{z}$ are ordered so that $\mathbf{x} \geq \mathbf{z}$, then under the influence of the same control input, $\mathbf{u}$ and $\mathbf{d}$, their corresponding next states are also ordered in a similar way, i.e., for the same set of $\mathbf{u}, \mathbf{d}$, $f(\mathbf{x}, \mathbf{u}, \mathbf{d}) \geq f(\mathbf{z}, \mathbf{u}, \mathbf{d})$. Inductively, their corresponding states maintain the same relationship as they evolve. In other words, if

  - $\mathbf{x}^0$ and $\mathbf{z}^0$ are two initial states satisfying $\mathbf{x}^0 \geq \mathbf{z}^0$, and
  - if their corresponding states evolve as $\mathbf{x}(t)$ and $\mathbf{z}(t)$ for $t > 0$,

  then for the same sequence of inputs, $\mathbf{u}(t)$ and disturbances, $\mathbf{d}(t)$, we have $\mathbf{x}(t) \geq \mathbf{z}(t)$.

- **Structural Property of Value Function:** If $\mathbf{x}, \mathbf{y} \in \mathcal{S}$ and $\mathbf{x} \geq \mathbf{y}$, then $V^*(\mathbf{x}) \leq V^*(\mathbf{y})$; furthermore, from the Evolution Invariance Property, $\mathbf{x}(t) \geq \mathbf{y}(t)$ for all $t \geq 0$. From the definition of partial ordering, the one-step reward at every t corresponding to initial condition $\mathbf{x}$ is no more than the one-step reward corresponding to the initial condition $\mathbf{y}$ for the same sequence of control actions $\mathbf{u}, \mathbf{d}$; correspondingly the total discounted reward associated with initial condition $\mathbf{x}$ is no more than than the corresponding quantity for initial condition $\mathbf{y}$. Taking expectation over all disturbances and maximizing over control actions, one gets $V^*(\mathbf{x}) \leq V^*(\mathbf{y})$; from the Evolution Invariance Property, $V^*(f(\mathbf{x}, \mathbf{u}, \mathbf{d})) \leq V^*(f(\mathbf{y}, \mathbf{u}, \mathbf{d}))$ for all $\mathbf{u}, \mathbf{d}$. This property implies that the value function, $\mathbf{V}^*$, satisfies linear inequalities describing the relationship between its components corresponding to states that can be compared.

- **Partitioning Property:** Assumptions 3 and 4 suggest the following definition of a partitioning scheme:

Definition (Partitioning Scheme): A partitioning scheme $\mathcal{P}$ of $\mathcal{S}$ and of cardinality M is defined to be a collection of M *disjoint* subsets of $\mathcal{S}$, i.e., $\mathcal{P} = \{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_M\}$ and

$$\cup_{k=1}^{M} \mathcal{S}_k = \mathcal{S},$$
$$\mathbf{x}_r = \mathbf{y}_r, \;\; \psi_s(\mathbf{x}_s) = \psi_s(\mathbf{y}_s), \;\; \forall \mathbf{x}, \mathbf{y} \in \mathcal{S}_k, \; k = 1, \ldots, M,$$
$$\mathcal{U}_{\mathbf{x}} = \mathcal{U}_{\mathbf{y}}, \forall \mathbf{x}, \mathbf{y} \in \mathcal{S}_k, \; k = 1, \ldots, M.$$

We will refer to the subsets $\mathcal{S}_k, \; k = 1, 2, \ldots, M$, as *partitions*. The one-step reward is only a function of a few state variables as specified in Assumption 4; in particular, for every $\mathbf{x} \in \mathcal{S}_i$ and for every $\mathbf{u}$, the one-step reward $r(\mathbf{x}, \mathbf{u})$ is the same and may be represented by $r_i(\mathbf{u})$ indicating its dependence only the index of the partition and the control input $\mathbf{u}$.

This problem also allows for the existence of partitions to be partially ordered in a manner that is consistent with the partial ordering of states. We define $\mathcal{S}_i \geq \mathcal{S}_j$ if and only if

1. for every $\mathbf{x} \in \mathcal{S}_i$, there is a $\mathbf{z} \in \mathcal{S}_j$ such that $\mathbf{x} \geq \mathbf{z}$; moreover, there is no $\mathbf{s} \in \mathcal{S}_j$ such that $\mathbf{s} \geq \mathbf{x}$, and
2. for every $\mathbf{z} \in \mathcal{S}_j$, there is a $\mathbf{x} \in \mathcal{S}_i$ such that $\mathbf{x} \geq \mathbf{z}$; moreover, there is no $\mathbf{s} \in \mathcal{S}_i$ such that $\mathbf{z} \geq \mathbf{s}$.

This problem allows for a partitioning scheme wherein if $\mathbf{x} \geq \mathbf{z}$, then one can find partitions, $\mathcal{S}_i \ni \mathbf{x}, \;\; \mathcal{S}_j \ni \mathbf{z}$ such that $\mathcal{S}_i \geq \mathcal{S}_j$.

We consider a piecewise constant approximation, i.e., the approximate value function is a constant over all the states in a given partition set. The determination of the approximate value function then reduces to finding the constants associated with the partition sets. We use the LP approach of (35) in this pursuit.

4.3 Properties of Generalized Bellman Inequalities

The solution to Bellman's equation can be expressed as as the following LP following (48), which we refer to as **OLP**: Let $\mathbf{c} \geq 0$ represent the probability distribution of initial states.

$$J = \min \mathbf{c} \cdot \mathbf{V}, \tag{4.7}$$
$$V(\mathbf{s}) \geq r(\mathbf{s}, \mathbf{u}) + \lambda \sum_{\mathbf{z} \in \mathcal{S}} p(\mathbf{s}, \mathbf{u}, \mathbf{z}) V(\mathbf{z}), \; \forall \mathbf{s} \in \mathcal{S}, \mathbf{u}. \tag{4.8}$$

The constraints (4.8) are referred to as Bellman Inequalities. From (35), any feasible $\mathbf{V}$ to the Bellman Inequalities upper bounds $\mathbf{V}^*$. However, a lower bounding procedure for $\mathbf{V}^*$ has received scant attention in the literature.

Let $\mathcal{P}_1, \ldots, \mathcal{P}_R$ be disjoint subsets of $\mathcal{S}$ and let their union be $\mathcal{S}$. These sets may not be partitions in the sense of the partioning scheme defined in the previous section. Let

$\geq$ denote the relationship between any two comparable states in the partial ordering; again, the partial ordering may be general and may not correspond to partial ordering defined earlier. In this chapter, we are interested in Generalized Bellman Inequalities, which incorporate the structure of value functions as follows:

1. **GBI-1:**

$$\begin{aligned} V(\mathbf{s}) &\geq r(\mathbf{s},\mathbf{u}) + \lambda \sum_{\mathbf{z}\in\mathcal{S}} p(\mathbf{s},\mathbf{u},\mathbf{z})V(\mathbf{z}),\ \forall \mathbf{s}\in\mathcal{S},\mathbf{u}, \\ V(\mathbf{x}) &\geq V(\mathbf{z}),\ \ \forall \mathbf{z}\geq\mathbf{x}. \end{aligned} \tag{4.9}$$

2. **GBI-2:**

$$\begin{aligned} \min_{\mathbf{x}\in\mathcal{P}_i} V(\mathbf{x}) &\geq \min_{\mathbf{x}\in\mathcal{P}_i}[r(\mathbf{x},\mathbf{u}) + \lambda \sum_{\mathbf{z}\in\mathcal{S}} p(\mathbf{x},\mathbf{u},\mathbf{z})V(\mathbf{z})],\ \ \forall i,\mathbf{u}, \\ V(\mathbf{x}) &= V(\mathbf{z}),\ \ \forall \mathbf{x},\mathbf{z}\in\mathcal{P}_i. \end{aligned} \tag{4.10}$$

Lemma 4.3.1 *The following properties hold for* **GBI-1***, and* **GBI-2***:*

1. *The inequalities are feasible and the feasible solutions are lower bounded.*
2. *If* $\mathbf{V}_1, \mathbf{V}_2$ *are two feasible solutions, then the componentwise minimum* $\min\{\mathbf{V}_1, \mathbf{V}_2\}$ *is also feasible.*
3. *The componentwise minimum of all feasible solutions,* $\bar{\mathbf{V}}_i := \min\{\mathbf{V} : \mathbf{V}$ *is feasible for* **GBI-i**$\}$*, is well defined and is also feasible. Hence, for every* $\mathbf{c}\geq 0$*, and any feasible* $\mathbf{V}$ *for* **GBI-i***,* $\mathbf{c}\cdot\mathbf{V} \geq \mathbf{c}\cdot\bar{\mathbf{V}}_i$*.*

Proof: If (1) holds, (2) and (3) readily follow. For $i = 1, 2$, let $\mathcal{F}_i$ denote the feasible set of **GBI-i**. Let $r_{max} = \max_{\mathbf{s},\mathbf{u}} r(\mathbf{s},\mathbf{u})$ and $r_0 := \min_{\mathbf{s},\mathbf{u}} r(\mathbf{s},\mathbf{u})$. Let $\mathbf{V}$ be a vector with every component of $\mathbf{V}$ being $\frac{r_{max}}{1-\lambda}$; it can readily be verified that **GBI-1** and **GBI-2** are feasible.

Let $\tilde{\mathcal{F}}_i$ correspond to the feasible set of **GBI-i** when the one step reward $r(\mathbf{s},\mathbf{u})$ is lowered to r_0. It follows that $\mathcal{F}_i \subset \tilde{\mathcal{F}}_i$ and hence, if the feasible solutions of $\tilde{\mathcal{F}}_i$ are lower bounded, then (1) is proved. It is easy to see that every component of the feasible solutions in $\tilde{\mathcal{F}}_i$ are lower bounded by $\frac{r_0}{1-\lambda}$ and hence (1) is proved. □

4.4 Main Results

In this section, we will present three theorems. The first two theorems deal with general MDPs, while the last theorem exploits the structure of the problem and partitioning schemes to simplify the computations. Theorem 1 deals with bounding the value function as given below:

Theorem 4.4.1 *For each* $i = 1, 2$*, let* $\bar{\mathbf{V}}_i$ *be the componentwise minimum of all feasible solutions of* **GBI-i***. Then,*

$$\bar{\mathbf{V}}_1 \geq \mathbf{V}^* \geq \bar{\mathbf{V}}_2.$$

Proof: Since $\bar{\mathbf{V}}_1$ satisfies Bellman Inequalities, $\bar{\mathbf{V}}_1 \geq \mathbf{V}^*$ from (35).

If we show that a lower bound, $\mathbf{V}_L$ of $\mathbf{V}^*$ is feasible for **GBI-2**, then by part (3) of Lemma 1, $\mathbf{V}_L \geq \bar{\mathbf{V}}_2$ as it is the componentwise minimum of all feasible solutions of **GBI-2**. It is easy to construct one such $\mathbf{V}_L$ as follows: for each $\mathbf{x} \in \mathcal{S}_i$, define $V_L(\mathbf{x}) = \min_{\mathbf{s}\in\mathcal{P}_i} V^*(\mathbf{s})$. Since $\mathbf{V}^*$ satisfies Bellman Inequalities, we have:

$$\begin{aligned}
\min_{\mathbf{s}\in\mathcal{P}_i} V^*(s) &\geq \min_{\mathbf{s}\in\mathcal{P}_i}[r(\mathbf{s},\mathbf{u}) + \lambda \sum_{\mathbf{z}\in\mathcal{S}} p(\mathbf{s},\mathbf{u},\mathbf{z})V^*(\mathbf{z})], \\
&\geq \min_{\mathbf{s}\in\mathcal{P}_i}[r(\mathbf{s},\mathbf{u}) + \lambda \sum_{\mathbf{z}\in\mathcal{S}} p(\mathbf{s},\mathbf{u},\mathbf{z})V_L(\mathbf{z})], \\
\Rightarrow \min_{\mathbf{s}\in\mathcal{P}_i} V_L(\mathbf{s}) &\geq \min_{\mathbf{s}\in\mathcal{P}_i}[r(\mathbf{s},\mathbf{u}) + \lambda \sum_{\mathbf{z}\in\mathcal{S}} p(\mathbf{s},\mathbf{u},\mathbf{z})V_L(\mathbf{z})].
\end{aligned}$$

Furthermore, for every $\mathbf{x}, \mathbf{s} \in \mathcal{P}_i$, by construction, $V_L(\mathbf{x}) = V_L(\mathbf{s})$. Hence, $\mathbf{V}_L$ is feasible for **GBI-2**. Since $\mathbf{V}^* \geq \mathbf{V}_L$, we have $\mathbf{V}^* \geq \bar{\mathbf{V}}_2$ by part (3) of Lemma 1. □

Remark 4 *The partial ordering of states in the second inequality constraint of* **GBI-1** *need not correspond to the partial ordering suggested by Assumption 4. If it does, the componentwise minimum of feasible solutions of* **GBI-1** *will indeed be* $\mathbf{V}^*$.

If, in any partition, any two states admit the same set of control actions (as the definition of partitioning scheme suggests), then one can employ the lower bound $\bar{\mathbf{V}}_2$ as an approximate value function in the construction of the following stationary sub-optimal policy: for every i and for every $\mathbf{x} \in \mathcal{P}_i$, define

$$u_{so}(\mathbf{x}) = arg\, max_{\mathbf{u}} \min_{\mathbf{s}\in\mathcal{P}_i}[r(\mathbf{s},\mathbf{u}) + \lambda \sum_{\mathbf{z}\in\mathcal{S}} p(\mathbf{s},\mathbf{u},\mathbf{z})\bar{V}_2(\mathbf{z})]. \tag{4.11}$$

In other words, the sub-optimal action is the same for every state in a partition; in such a case, if there is a simple rule to determine the membership of a state $\mathbf{x}$ to a partition $\mathcal{P}_i$, one can implement the above specified stationary, sub-optimal policy easily by keeping track of the control action associated with each partition.

Associated with the sub-optimal policy, let $P_{so}, \mathbf{r}_{so}, \mathbf{V}_{so}$ be the respective probability transition matrix, reward vector and the sub-optimal value function (or simply performance function). The entry of P_{so} in the row corresponding to $\mathbf{x}$ and column corresponding to $\mathbf{z}$ is given by $p(\mathbf{x}, u_{so}(\mathbf{x}), \mathbf{z})$; the one-step reward corresponding to state $\mathbf{x}$ is $r(\mathbf{x}, u_{so}(\mathbf{x}))$. One may also note that

$$\mathbf{V}_{so} = \mathbf{r}_{so} + \lambda P_{so}\mathbf{V}_{so}.$$

The following theorem relates the performance $\mathbf{V}_{so}$ of the sub-optimal policy to the lower bound $\bar{\mathbf{V}}_2$ of the value function, $\mathbf{V}^*$:

Theorem 4.4.2

$$\mathbf{V}^* \geq \mathbf{V}_{so} \geq \bar{\mathbf{V}}_2.$$

Proof: Let $\mathbf{c} > 0$ and consider the problem of minimizing $\mathbf{c} \cdot \mathbf{V}$ subject to $\mathbf{V}$ satisfying the constraints of **GBI-2**. By part (3) of Lemma 1, $\bar{\mathbf{V}}_2$ is *the* optimal solution and by part (1) of Lemma 1, it is lower bounded.

For every i, there is a $\mathbf{u}_{so}(i)$ for which the disjunctive inequality is tight, i.e.,

$$\min_{\mathbf{x}\in\mathcal{P}_i} \bar{V}_2(\mathbf{x}) = \min_{\mathbf{x}\in\mathcal{P}_i}[r(\mathbf{x},\mathbf{u}_{so}) + \lambda \sum_{\mathbf{z}\in\mathcal{S}} p(\mathbf{x},\mathbf{u}_{so},\mathbf{z})\bar{V}_2(\mathbf{z})], \quad \forall i,$$

and

$$\begin{aligned}\min_{\mathbf{x}\in\mathcal{P}_i}[r(\mathbf{x},\mathbf{u}_{so}) + \lambda \sum_{\mathbf{z}\in\mathcal{S}} p(\mathbf{x},\mathbf{u}_{so},\mathbf{z})\bar{V}_2(\mathbf{z})] \geq \\ \min_{\mathbf{x}\in\mathcal{P}_i}[r(\mathbf{x},\mathbf{u}) + \lambda \sum_{\mathbf{z}\in\mathcal{S}} p(\mathbf{x},\mathbf{u},\mathbf{z})\bar{V}_2(\mathbf{z})], \quad \forall \mathbf{u};\end{aligned}$$

Otherwise, if no such $\mathbf{u}_{so}(i)$ exists, then one may drop all the non-binding constraints associated with $\min_{\mathbf{x}\in\mathcal{P}_i} V(\mathbf{x})$ in **GBI-2** and yet, the optimal solution will not change. In such a case, the term $\min_{\mathbf{x}\in\mathcal{P}_i} V(\mathbf{x})$ will not be lower bounded as there will be no associated lower bounding constraint and $\min_{\mathbf{x}\in\mathcal{P}_i} \bar{\mathbf{V}}_2(\mathbf{x})$ will not be lower bounded; this is contrary to part (1) of Lemma 1.

Since the optimal solution does not change by dropping all non-binding constraints, we note that $\bar{\mathbf{V}}_2$ will remain optimal for the following Disjunctive LP (DLP):

$$\begin{aligned} J &= \min \mathbf{c} \cdot \mathbf{V}, \\ \min_{\mathbf{x}\in\mathcal{P}_i} V(\mathbf{x}) &\geq \min_{\mathbf{x}\in\mathcal{P}_i}[r(\mathbf{x},\mathbf{u}_{so}) + \lambda \sum_{\mathbf{z}\in\mathcal{S}} p(\mathbf{x},\mathbf{u}_{so}(i),\mathbf{z})V(\mathbf{z})], \quad \forall i, \\ V(\mathbf{x}) &= V(\mathbf{z}), \quad \forall i, \ \ \mathbf{x},\mathbf{z} \in \mathcal{P}_i. \end{aligned}$$

One may consider the constraints of the above DLP as a special case of **GBI-2** by restricting the set of control actions at every state. By Lemma 1, $\bar{\mathbf{V}}_2$ is the componentwise minimum of the solutions of this DLP also; hence, it suffices to show that a lower bound for $\mathbf{V}_{so}$ is feasible for the above DLP. Indeed, we may construct a lower bound, $\mathbf{V}_{lb}$ for $\mathbf{V}_{so}$ as follows: for each $\mathbf{x} \in \mathcal{S}_i$, define $V_{lb}(\mathbf{x}) := \min_{\mathbf{s}\in\mathcal{S}_i} V_{so}(\mathbf{s})$. Clearly, by construction, $\mathbf{V}_{so} \geq \mathbf{V}_{lb}$. It is sufficient to show that $\mathbf{V}_{lb}$ is feasible to the above DLP.

For all $\mathbf{x} \in \mathcal{P}_i$, we note that

$$\begin{aligned} V_{so}(\mathbf{x}) = r(\mathbf{x},\mathbf{u}_{so}(i)) + \lambda \sum_{\mathbf{z}\in\mathcal{S}} p(\mathbf{x},\mathbf{u}_{so}(i),\mathbf{z})V_{so}(\mathbf{z}), \quad \forall i, \\ \Rightarrow \min_{\mathbf{x}\in\mathcal{P}_i} V_{lb}(\mathbf{x}) = \min_{\mathbf{x}\in\mathcal{P}_i} V_{so}(\mathbf{x}) \\ \geq \min_{\mathbf{x}\in\mathcal{P}_i}[r(\mathbf{x},\mathbf{u}_{so}(i)) + \lambda \sum_{\mathbf{z}\in\mathcal{S}} p(\mathbf{x},\mathbf{u}_{so}(i)V_{lb}(\mathbf{z})], \quad \forall i. \end{aligned}$$

Since $V_{lb}(\mathbf{x}) = V_{lb}(\mathbf{z})$ for all $\mathbf{x},\mathbf{z} \in \mathcal{P}_i$, it follows that V_{lb} satisfies the constraints of the DLP. Hence, $\mathbf{V}_{so} \geq \mathbf{V}_{lb} \geq \bar{\mathbf{V}}_2$. $\square$

4.4.1 Exploiting the structure of the problem

In order to bound the value function $\mathbf{V}^*$ and guarantee the performance of the suboptimal policy, one must compute $\bar{\mathbf{V}}_1$ and $\bar{\mathbf{V}}_2$. In general, it is a difficult proposition. Exploiting the structure of the problem and partitioning facilitates the computation of upper and lower bounds. The structure of the problem has thus far not been exploited and the properties of Theorems 1 and 2 hold for general MDPs. From hereon, we will exploit the structure of the problem to simplify the computation of upper and lower bounds.

Let $\mathcal{S}_1, \ldots, \mathcal{S}_M$ be partitions of $\mathcal{S}$. If, for some integer $j \in [1, M]$, $\mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{d}_l) \in \mathcal{S}_j$, then we will define $\bar{f}(\mathbf{x}, \mathbf{u}, \mathbf{d}_l) = j$.

There are two kinds of partitions that one can encounter in this application depending on the model of incursions. If one allows for incursions to occur independently across all the UGS stations, then every partition will have a maximum and minimum element. On the other hand, if one allows for only one incursion at a time across all the UGS stations, then every partition will have a few maximal and minimal states. For this reason, we will deal with these two partitioning schemes; we also observe that in both cases, the one-step reward obeys the following structure owing to Assumption 4 and the partitioning property: for every i and for every $\mathbf{x} \in \mathcal{S}_i$, we have $r(\mathbf{x}, \mathbf{u}) = r_i(\mathbf{u})$ for every $\mathbf{u}$.

Theorem 4.4.3 *Suppose for every i, let $\mathcal{S}_{min,i}$, $\mathcal{S}_{max,i}$ be respectively the set of minimal and maximal elements of $\mathcal{S}_i$. Let $\mathbf{c}$ be any positive vector. Then,*

1. *an upper bound $\bar{\mathbf{V}}_{ub}$ of $\mathbf{V}^*$ may be computed from the optimal solution, $\bar{\mathbf{w}}$ following upper bounding LP referred to as* **UBLP***:*

$$\begin{aligned} J_u &= \min \sum_{j=1}^{M} [\sum_{\mathbf{x} \in \mathcal{S}_i} c(\mathbf{x})] w(i), \\ w(i) &\geq r_i(\mathbf{u}) + \lambda \sum_{l=1}^{L} p_l w(\bar{f}(\mathbf{z}, \mathbf{u}, \mathbf{d}_l)), \ \forall i, \mathbf{u}, \mathbf{z} \in \mathcal{S}_{min,i}, \\ w(j) &\geq w(i), \ \ \forall \mathcal{S}_i \geq \mathcal{S}_j. \end{aligned}$$

   and $\bar{V}_{ub}(\mathbf{x}) = \bar{w}(i)$ for all $\mathbf{x} \in \mathcal{S}_i$.

2. *A lower bound $\bar{\mathbf{V}}_2$ of $\mathbf{V}^*$ can be computed from the optimal value, $\bar{\mathbf{w}}$ of the following lower bounding disjunctive LP referred to as* **LBDLP***:*

$$\begin{aligned} J_l &= \sum_{j=1}^{M} [\sum_{\mathbf{x} \in \mathcal{S}_i} c(\mathbf{x})] w(i), , \\ w(i) &\geq \min_{\mathbf{z} \in \mathcal{S}_{max,i}} [r_i(\mathbf{u}) + \lambda \sum_{l=1}^{L} p_l w(\bar{f}(\mathbf{z}, \mathbf{u}, \mathbf{d}_l))], \ \ \forall i, \mathbf{u}, \end{aligned}$$

   and $\bar{V}_2(\mathbf{x}) = \bar{w}(i)$ for every $\mathbf{x} \in \mathcal{S}_i$.

Note that the **UBLP** and **LBDLP** have fewer variables in $w(i),\ i = 1, \ldots, M$ with the number of inequality constraints being correspondingly smaller. Hence, these two programs are more tractable. We now return to the proof of Theorem (4.4.3):
Proof: Using Assumption 2 and the assumption that $r(\mathbf{x}, \mathbf{u}) = r_i(\mathbf{u})$ for every $\mathbf{x} \in \mathcal{S}_i$, we may express the Bellman Inequalities as:

$$V(\mathbf{x}) \geq r_i(\mathbf{u}) + \lambda \sum_{l=1}^{L} p_l V(f(\mathbf{x}, \mathbf{u}, \mathbf{d}_l)), \quad \forall i, \mathbf{u},\ \mathbf{x} \in \mathcal{S}_i. \tag{4.12}$$

1. Consider the following minimization problem:

$$\begin{aligned} J_s &= \min \mathbf{c} \cdot \mathbf{V}, && (4.13)\\ V(\mathbf{x}) &\geq r_i(\mathbf{u}) + \lambda \sum_{l=1}^{L} p_l V(f(\mathbf{x}, \mathbf{u}, \mathbf{d}_l)), && (4.14)\\ V(\mathbf{x}) &\geq V(\mathbf{z}), \quad \forall \mathbf{z} \geq \mathbf{x}. && (4.15) \end{aligned}$$

   Since $\mathbf{x} \geq \mathbf{z}$ for some $\mathbf{z} \in \mathcal{S}_{min,i}$, if $\mathbf{V}$ satisfies the following strengthened version of Bellman inequality for all $\mathbf{x} \in \mathcal{S}_i,\ \ \mathbf{z} \in \mathcal{S}_{min,i}$,

$$V(\mathbf{x}) \geq r_i(\mathbf{u}) + \lambda \sum_{l=1}^{L} p_l V(f(\mathbf{z}, \mathbf{u}, \mathbf{d}_l)),$$

   it would automatically satisfy Bellman's inequalities:

$$V(\mathbf{x}) \geq r_i(\mathbf{u}) + \lambda \sum_{l=1}^{L} p_l V(f(\mathbf{x}, \mathbf{u}, \mathbf{d}_l)), \ \ \forall \mathbf{x} \in \mathcal{S}_i.$$

   By part (3) of Lemma 1 applied to **GBI-1**, the optimal solution, $\bar{\mathbf{V}}$, of the following LP dominates $\mathbf{V}^*$:

$$\begin{aligned} J &= \min \mathbf{c} \cdot \mathbf{V},\\ V(\mathbf{x}) &\geq r_i(\mathbf{u}) + \lambda \sum_{l=1}^{L} p_l V(f(\mathbf{z}, \mathbf{u}, \mathbf{d}_l)), \ \ \forall \mathbf{x} \in \mathcal{S}_i,\ \mathbf{z} \in \mathcal{S}_{min,i},\\ V(\mathbf{z}) &\geq V(\mathbf{x}), \quad \forall \mathbf{x} \geq \mathbf{z}, \quad \mathbf{x}, \mathbf{z} \in \mathcal{S}. \end{aligned}$$

   This LP and **UBLP** have the same optimal value and the optimal solution of one can be used to construct the optimal solution of the other in the following way:

   - Since $\bar{V}_{ub}(\mathbf{x}) = \bar{w}(i)$ for all $\mathbf{x} \in \mathcal{S}_i$, we can see that $\bar{\mathbf{V}}_{ub}$ readily satisfies Bellman's inequalities of the above LP. The last set of constraints is met by virtue of the property of partitioning: if $\mathbf{x} \geq \mathbf{z}$, then there exist partitions $\mathcal{S}_i \ni \mathbf{x}, \quad \mathcal{S}_j \ni \mathbf{z}$ such that $\mathcal{S}_i \geq \mathcal{S}_j$. Since $\bar{w}(j) \geq \bar{w}(i)$ for all $\mathcal{S}_i \geq \mathcal{S}_j$, it follows that $\bar{V}_{ub}(\mathbf{z}) = \bar{w}(j) \geq \bar{w}(i) = \bar{V}_{ub}(\mathbf{x})$. Since $\bar{\mathbf{V}}$ is optimal, $\mathbf{c} \cdot \bar{\mathbf{V}} \leq \mathbf{c} \cdot \bar{\mathbf{V}}_{ub} = \sum_{i=1}^{M} [\sum_{\mathbf{x} \in \mathcal{S}_i} c(\mathbf{x})] \bar{w}(i)$. By property (3) of Lemma 1, we additionally have $\bar{\mathbf{V}}_{ub} \geq \bar{\mathbf{V}}$.

- By the same token, if we set $w(i) = \bar{V}(\mathbf{x}),\ \forall x \in \mathcal{S}_i$, we see that it is feasible for **UBLP**. Hence, $\sum_{i=1}^{M}[\sum_{\mathbf{x}\in\mathcal{S}_i} c(\mathbf{x})]\bar{w}(i) \leq \sum_{i=1}^{M}[\sum_{\mathbf{x}\in\mathcal{S}_i} c(\mathbf{x})]w(i) = \mathbf{c}\cdot\bar{\mathbf{V}}$.

Since $\mathbf{c} > 0$ and $\bar{\mathbf{V}}_{ub} \geq \bar{\mathbf{V}}$, $\mathbf{c}\cdot\bar{\mathbf{V}} = \mathbf{c}\cdot\bar{\mathbf{V}}_{ub} \Rightarrow \bar{\mathbf{V}} = \bar{\mathbf{V}}_{ub}$. Since $\bar{\mathbf{V}} \geq \mathbf{V}^*$, it follows that $\bar{\mathbf{V}}_{ub} \geq \mathbf{V}^*$.

2. For every $\mathbf{x} \in \mathcal{S}_i$, $\mathbf{z} \geq \mathbf{x}$ for some $\mathbf{z} \in \mathcal{S}_{max,i}$; it follows by Lemma 3 that $V^*(\mathbf{x}) \geq V^*(\mathbf{z})$. Let us define $w^*(i) := \min_{\mathbf{x}\in\mathcal{S}_i} V^*(\mathbf{x}) = \min_{\mathbf{z}\in\mathcal{S}_{max,i}} V^*(\mathbf{z})$. Since $\mathbf{V}^*$ satisfies Bellman inequalities, it follows that

$$\begin{aligned} w^*(i) &= \min_{\mathbf{x}\in\mathcal{S}_i} V^*(\mathbf{x}) = \min_{\mathbf{z}\in\mathcal{S}_{max,i}} V^*(\mathbf{z}) \\ &\geq \min_{\mathbf{z}\in\mathcal{S}_{max,i}} [r_i(\mathbf{u}) + \lambda \sum_{l=1}^{L} p_l V^*(f(\mathbf{z},\mathbf{u},\mathbf{d}_l))],\ \forall i,\mathbf{u}. \end{aligned}$$

Clearly, then $(\mathbf{w}^*, \mathbf{V}^*)$ satisfy the following constraints of the following DLP:

$$\begin{aligned} J_l &= \min \mathbf{c}\cdot\mathbf{V}, \\ w(i) &\geq \min_{\mathbf{z}\in\mathcal{S}_{max,i}} [r_i(\mathbf{u}) + \lambda \sum_{l=1}^{L} p_l w(\bar{f}(\mathbf{z},\mathbf{u},\mathbf{d}_l))],\ \ \forall i,\mathbf{u}, \\ V(\mathbf{x}) &\geq w(i), \quad \forall \mathbf{x} \in \mathcal{S}_i,\ \forall i. \end{aligned}$$

The constraints of the above DLP are of the form **GBI-2** and the conclusions of Lemma 1 hold for this reason. Since $\mathbf{c} > 0$, by Lemma 1, the optimal solution of the above DLP, say $(\mathbf{w}_{lb}, \mathbf{V}_{lb})$ is the componentwise minimum of all its feasible solutions and hence, $\mathbf{V}^* \geq \mathbf{V}_{lb}$. Since $\mathbf{c} > 0$, the last inequality becomes binding at optimum, and the optimal solution of the above DLP $(\mathbf{w}_{lb}, \mathbf{V}_{lb})$ is feasible for **LBDLP**; similarly, the optimal solution $(\bar{\mathbf{w}}, \bar{\mathbf{V}}_2)$ of **LBDLP** is feasible for the above DLP. Hence, $\bar{\mathbf{V}}_2 = \mathbf{V}_{lb} \leq \mathbf{V}^*$. □

Remark 5

- *The partitioning property is required to reduce the number of constraints for computing the upper bound from* **GBI-1** *to* **UBLP**.
- *If the model of incursions allows for their independent occurrence at each of the UGS stations, then the cardinality of the sets $\mathcal{S}_{max,i}$ and $\mathcal{S}_{min,i}$ is* 1*, thereby reducing* **LBDLP** *to an LP. This model is assumed in the numerical simulations.*
- *Even if $\mathcal{S}_{max,i}$ were to be of higher cardinality, it is possible that the DLP constraint may reduce to a linear constraint; for example, $\mathbf{z}_1, \mathbf{z}_2 \in \mathcal{S}_{max,i}$ are distinct and $\bar{f}(\mathbf{z}_1,\mathbf{u},\mathbf{d}_l) = \bar{f}(\mathbf{z}_2,\mathbf{u},\mathbf{d}_l)$ for every l, such a simplification can occur. This situation corresponds to states in the set $\mathcal{S}_{max,i}$ transitioning to the same set of partitions for every $\mathbf{u}$ and $\mathbf{d}$.*

- *The solution of DLP listed can be iteratively found as follows:*
  - **Step 0:** *Set $k = 0$ and pick $\mathbf{z}_i^0 \in \mathcal{S}_{max,i}$ arbitrarily or by using upper bound in Step 2 below for $\mathbf{w}^0$.*
  - **Step 1:** *Solve the LP:*

$$\begin{aligned} J_k &= \sum_{j=1}^{M}[\sum_{\mathbf{x}\in\mathcal{S}_i} c(\mathbf{x})]w(i), \\ w(i) &\geq r_i(\mathbf{u}) + \lambda \sum_{l=1}^{L} p_l w(\bar{f}(\mathbf{z}_i^k, \mathbf{u}, \mathbf{d}_l))], \ \forall i, \mathbf{u}. \end{aligned}$$

  - **Step 2:** *If $\mathbf{w}^k$ is the optimal solution at the k^{th} iteration, one can then compute*

$$\mathbf{z}_i^{k+1} = arg\, min_{\mathbf{z}\in\mathcal{S}_{max,i}}[r_i(\mathbf{u}) + \lambda \sum_{l=1}^{L} p_l w^k(\bar{f}(\mathbf{z}, \mathbf{u}, \mathbf{d}_l))].$$

  - **Step 3:** *If $\mathbf{z}_i^{k+1} = \mathbf{z}_i^k$, stop; else, set $k = k + 1$ and go to Step 1.*

  At each iteration, the optimal value of LP drops as the optimal solution for the k^{th} iteration is feasible for the $(k+1)^{st}$ iteration, and hence, this algorithm will obtain the optimal solution of DLP in a finite number of iterations. The solution of LP in Step 1 is computationally tractable as the number of variables and constraints are only of the order of the number of partitions. The step 2 hides the complexity - if $\mathcal{S}_{max,i}$ is of low cardinality, this is computationally tractable. otherwise, it will be difficult to solve.

- *The performance guarantee of a suboptimal policy constructed using $\bar{\mathbf{V}}_2$ as in Theorem 2 still holds.*

In the following section, we provide numerical examples that corroborate the main results.

4.5 Illustrative Example

Consider a perimeter to be monitored with the aid of $n_r = 2$ identical robots. Let $N = 8$ nodes discretize the perimeter uniformly, with some of the nodes corresponding to the locations of UGS. Let the set of nodes be labeled as $\mathcal{N} := \{0, 1, \ldots, N-1\}$ and let the set of UGS locations be $\Omega \subset \mathcal{N}$ ($\Omega = \{0, 2, 4, 6\}$), in this case, the stations are symmetricially located. Note that $|\Omega| = n_s$. If a UGS detects an incursion, an alert is raised at the location and communicated instantaneously to the robots. Let $a_i(t)$ denote the action of the i^{th} robot at time t, so that the control input $\mathbf{u}(t) = (a_1(t), a_2(t), \ldots, a_{n_r}(t))$. The set of allowable actions for the robot are $\{1, 0, -1\}$ with $a_i(t) = 0$ if it dwells at its current location and equals 1 or -1 respectively if it moves counterclockwise or clockwise. The

maximum number of allowable values of $\mathbf{u}$ is 3^{n_r}. The disturbance input (incursion) at the j^{th} station be denoted by $d_j(t) \in \{0, 1\}$, with $d_j(t) = 1$ if there is an incursion at time t and is 0 otherwise. The disturbance input, $\mathbf{d}(t)$ is an n_s-tuple with its j^{th} component being $d_j(t),\ j \in \Omega$. Let $\delta(\cdot)$ denote the Kronecker delta function and $\bar{\delta}(\cdot) = 1 - \delta(\cdot)$. We will assume that each station has an independent alert queue so that the maximum number of allowable values of the disturbance input $\mathbf{d}(t)$ is 2^{n_s}, and the arrival of incursions at each queue is a stochastic process. Let the probability of no incursions occurring at a station in a time unit be p_α. Then probability that k stations raise an alert at each time step is $p_\alpha^{n_s-k}(1-p_\alpha)^k$. For simplication, if a new alert arises at a station which already has an alert, then the new alert will be processed with the old alert.

We consider the following additional restriction on the motion of the robots: A robot can only dwell at a UGS location; hence, the allowable actions at a non-UGS location for the i^{th} robot is $\{-1, 1\}$.

Let $l_i(t), T_i(t)$ respectively denote the current location of the i^{th} robot and the time it has spent at its current location. Let $l_s(t)$ denote a distance to the nearest station from the first robot in CCW, and $l_r(t)$ be distance from the first robot to the second one in CCW. The state of the robots is given by $\mathbf{x}_r(t) = (l_s(t), l_r(t), T_1(t), T_2(t))$. Let $\tau_j(t)$ denote the time elapsed since an alert, that is yet to be serviced, was raised at the j^{th} station in CCW. The state $\mathbf{x}_s(t)$ is the n_s-tuple of time delays, with the j^{th} component being the time delay $\tau_j(t)$.

The governing equations for this example may be expressed as:

$$l_s(t+1) = (l_s(t) + a_1(t)) \mod N/n_s, \tag{4.16}$$

$$l_r(t+1) = (l_r(t) + a_2(t)) \mod N, \tag{4.17}$$

$$T_i(t+1) = (T_i(t) + 1)\delta(a_i(t)), \quad i = 1, 2, \tag{4.18}$$

$$\tau_j\,(t+1) = h(\tau_j(t), l_1(t), a_1(t), l_2(t), a_2(t), d_j(t)), \forall j \in \Omega, \tag{4.19}$$

where,

$$h(\tau_j, l_1, a_1, l_2, a_2, d_j) :=$$
$$\max\left\{(\tau_j + 1)\sigma\left(\tau_j(t)\right)\left[1 - \max_{i=1,2}\left\{\delta(l_i - j)\delta(a_i)\right\}\right], d_j\right\}$$

The one step reward function is given by,

$$r(\mathbf{x}, \mathbf{u}) = \sum_{k=1}^{K} \psi_r(T_k) - \beta||\mathbf{x}_s||_\infty. \tag{4.20}$$

The information gain function, $\psi_r(\cdot)$ as a function monotonically increasing in the time spent by the robot at an UGS location (for details on the operator modeling - see (44)). The second term penalizes the tardy response of UAVs.

The other parameters were chosen to be: $p_a = e^{-2/60}$, weighing factor, $\beta = 0.01$ and discount factor, $\lambda = 0.9$. The corresponding MDP has 1,395,456 states. To reduce the size of problem, states with same $\mathbf{x}_r$, worst service delay, $\bar{\tau}(\mathbf{x}_s(t)) := \max_{j\in\Omega} \tau_j(t)$, and alert status, $A(\mathbf{x}_s(t)) := (\bar{\delta}(\tau_1(t)), \ldots, \bar{\delta}(\tau_{n_s}(t))$, are aggregated in the same partition. In

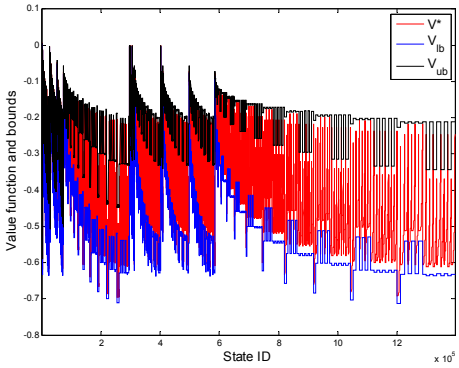


Figure 4.1: Value function and bounds (all states)

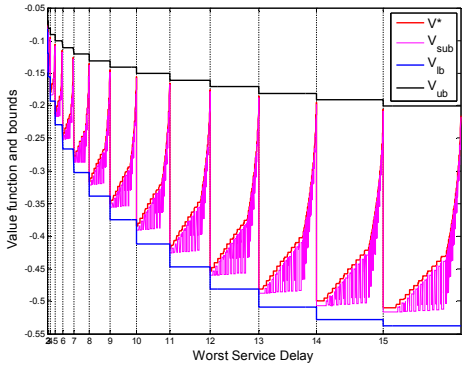


Figure 4.2: Value function and bounds (Sampled Partitions): $l_s = 0, l_r = 4, T_1 = 0, T_2 = 0, A = (0, 1, 1, 1)$

this partition, $r(\mathbf{x}, \mathbf{u}) = r(\mathbf{y}, \mathbf{u}), \forall \mathbf{x}, \mathbf{y} \in \mathcal{S}_i, \forall \mathbf{u}$, and the number of partitions is 15,546. Figure 4.1 and 4.2 show the value function and its bounds. The optimal value function, $\mathbf{V}^*$, is found from the value iteration. An upper bound, $\mathbf{V}_{ub}$, and a lower bound, $\mathbf{V}_{lb}$, are solutions to **UBLP** and the iterative LP from Remark 2. Another value function, $\mathbf{V}_{sub}$, represents the suboptimal performance value function which is the actual value function when the suboptimal policy applied to the original MDP. The boundness of the value function is shown in Fig. 4.1 that shows the value functions and bounds for all states. Since there are too many data in one plot, we sampled some states so that it is readable. We pick states which belong to partitions with $l_s = 0, l_r = 4, T_1 = 0, T_2 = 0, A = (0, 1, 1, 1)$, and all $\bar{\tau}$s. In Fig. 4.2, the dotted lines represent seperation of partitions. In this data set, only difference between partitions is $\bar{\tau}$ shown in $X-$axis. As we can see, the value function and the suboptimal performance value function are bounded by the upper and lower bounds. We can also see that the suboptimal performance value function is very close to the value function. In this example, percentage error between upper and lower bounds is 57.5%, and one between $\mathbf{V}^*$ and $\mathbf{V}_{sub}$ is 4.2%.

Bibliography

[1] V. V. Vazirani, *Approximation algorithms*. Springer Verlag, 2001.

[2] N. Christofides, "Worst-case analysis of a new heuristic for the travelling salesman problem." DTIC Document, Tech. Rep., 1976.

[3] S. Rathinam and R. Sengupta, "Lower and upper bounds for a multiple depot UAV routing problem," in *Decision and Control, 2006 45th IEEE Conference on*, 2006, pp. 5287–5292.

[4] S. Rathinam, R. Sengupta, and S. Darbha, "A resource allocation algorithm for multivehicle systems with nonholonomic constraints," *IEEE Transactions Automation Science and Engineering*, vol. 4, pp. 98–104, 2007.

[5] M. Grtschel, L. Lovsz, and A. Schrijver, "The ellipsoid method and its consequences in combinatorial optimization," *Combinatorica*, vol. 1, pp. 169–197, 1981, 10.1007/BF02579273.

[6] D. Williamson, "Analysis of the held-karp heuristic for the traveling salesman problem," Ph.D. dissertation, Master Thesis, MIT, 1990.

[7] M. Goemans and D. Bertsimas, "Survivable networks, linear programming relaxations and the parsimonious property," *Mathematical Programming*, vol. 60, no. 1-3, pp. 145–166, 1993. [Online]. Available: http://dx.doi.org/10.1007/BF01580607

[8] T. Zajkowski, S. Dunagan, and J. Eilers, "Small UAS communications mission," in *Eleventh Biennial USDA Forest Service Remote Sensing Applications Conference, Salt Lake City, UT*, 2006.

[9] E. W. Frew and T. X. Brown, "Networking issues for small unmanned aircraft systems," *Unmanned Aircraft Systems*, pp. 21–37, 2009.

[10] J. A. Curry, J. Maslanik, G. Holland, J. Pinto, G. Tyrrell, and J. Inoue, "Applications of aerosondes in the arctic," *Bull. Am. Meteorol. Soc*, vol. 85, no. 12, pp. 1855–1861, 2004.

[11] "NOAA and partners conduct first successful unmanned aircraft hurricane observation by flying through ophelia." [Online]. Available: http://www.noaanews.noaa.gov/stories2005/s2508.htm

[12] C. E. Corrigan, G. C. Roberts, M. V. Ramana, D. Kim, V. Ramanathan *et al.*, "Capturing vertical profiles of aerosols and black carbon over the indian ocean using autonomous unmanned aerial vehicles," *Atmospheric Chemistry and Physics*, vol. 8, no. 3, pp. 737–747, 2008.

[13] "Soldiers train with raven UAV's, united states army." [Online]. Available: http://www.army.mil/article/5644/soldiers-train-with-raven-uavs/

[14] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of Mathematics*, vol. 79, no. 3, pp. 497–516, Jul. 1957, ArticleType: research-article / Full publication date: Jul., 1957 / Copyright 1957 The Johns Hopkins University Press.

[15] S. Yadlapalli, W. Malik, S. Darbha, and M. Pachter, "A lagrangian-based algorithm for a multiple depot, multiple traveling salesmen problem," *Nonlinear Analysis: Real World Applications*, vol. 10, no. 4, pp. 1990 – 1999, 2009.

[16] G. L. Feithans, A. J. Rowe, J. E. Davis, M. Holland, and L. Berger, "Vigilant spirit control station (vscs)- "the face of counter"," *Proceedings of the AIAA Guidance, Navigation and Control Conference and Exhibition, Honululu, Hawaii*, no. AIAA Paper Number: 2008-6309, August 2008.

[17] P. Oberlin, S. Rathinam, and S. Darbha, "Today's Traveling Salesman Problem," *IEEE Robotics & Automation Magazine*, vol. 17, no. 4, pp. 70–77, Dec. 2010. [Online]. Available: http://dx.doi.org/10.1109/MRA.2010.938844

[18] A. M. Frieze, G. Galbiati, and F. Maffioli, "On the worst-case performance of some algorithms for the asymmetric traveling salesman problem," *Networks*, vol. 12, no. 1, pp. 23–39, 1982.

[19] H. Kaplan, M. Lewenstein, N. Shafrir, and M. Sviridenko, "Approximation algorithms for asymmetric TSP by decomposing directed regular multigraphs," *Journal of the ACM (JACM)*, vol. 52, no. 4, pp. 602–626, 2005.

[20] M. Blaser, "A new approximation algorithm for the asymmetric TSP with triangle inequality," *ACM Transactions on Algorithms (TALG)*, vol. 4, no. 4, p. 47, 2008.

[21] S. Khuller, A. Malekian, and J. Mestre, "To fill or not to fill: The gas station problem," in *Algorithms ESA 2007*, L. Arge, M. Hoffmann, and E. Welzl, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, vol. 4698, pp. 534–545.

[22] P. B. Sujit and D. Ghose, "Two-agent cooperative search using game models with endurance-time constraints," *Engineering Optimization*, vol. 42, no. 7, pp. 617–639, 2010.

[23] G. Ghiani, F. Guerriero, G. Laporte, and R. Musmanno, "Tabu search heuristics for the arc routing problem with intermediate facilities under capacity and length

restrictions," *Journal of Mathematical Modelling and Algorithms*, vol. 3, no. 3, pp. 209–223, 2004.

[24] M. Polacek, K. F. Doerner, R. F. Hartl, and V. Maniezzo, "A variable neighborhood search for the capacitated arc routing problem with intermediate facilities," *Journal of Heuristics*, vol. 14, no. 5, pp. 405–423, 2008.

[25] E. Angelelli and M. Grazia Speranza, "The periodic vehicle routing problem with intermediate facilities," *European Journal of Operational Research*, vol. 137, no. 2, pp. 233–247, 2002.

[26] B. Crevier, J. F. Cordeau, and G. Laporte, "The multi-depot vehicle routing problem with inter-depot routes," *European Journal of Operational Research*, vol. 176, no. 2, pp. 756–773, 2007.

[27] E. D. Taillard, G. Laporte, and M. Gendreau, "Vehicle routeing with multiple use of vehicles," *The Journal of the Operational Research Society*, vol. 47, no. 8, pp. pp. 1065–1070, 1996.

[28] Q. H. Zhao, S. Y. Wang, K. K. Lai, and G. Xia, "A vehicle routing problem with multiple use of vehicles," *Advanced Modeling and Optimization*, vol. 4, no. 3, pp. 21–40, 2002.

[29] J. O. Royset, W. M. Carlyle, and R. K. Wood, "Routing military aircraft with a constrained shortest-path algorithm," *Military Operations Research*, vol. 14, no. 3, pp. 31–52, 2009.

[30] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[31] S.-H. Lin, "Finding optimal refueling policies in transportation networks," in *Algorithmic Aspects in Information and Management*, R. Fleischer and J. Xu, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, vol. 5034, pp. 280–291.

[32] "IBM - ILOG: CPLEX optimization studio 12.2." [Online]. Available: http://www.ilog.com/products/cplex

[33] "Python programming language (v.2.7.2)." [Online]. Available: http://www.python.org/psf/

[34] T. L. Magnanti and L. A. Wolsey, "Optimal trees," *Handbooks in operations research and management science*, vol. 7, pp. 503–615, 1995.

[35] P. J. Schweitzer and A. Seidmann, "Generalized polynomial approximations in Markovian decision processes," *Journal of Mathematical Analysis and Applications*, vol. 110, no. 2, pp. 568–582, 1985.

[36] M. Trick and S. Zin, "Spline approximation to value functions: A linear programming approach," *Macroeconomic Dynamics*, vol. 1, pp. 255–277, 1997.

[37] B. Van Roy, "Performance loss bounds for approximate value iteration with state aggregation," *Mathematics of Operations Research*, vol. 31, no. 2, pp. 234–244, 2006.

[38] D. Kingston, R. Beard, and R. Holt, "Decentralized perimeter surveillance using a team of UAVs," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1394–1404, 2008.

[39] D. A. Paley, L. Techy, and C. A. Woolsey, "Coordinated perimeter patrol with minimum-time alert response," in *Proc. Guidance, Navigation and Control Conf.*, no. AIAA 2009-6210, Chicago, IL, 2009.

[40] J. Marier, C. Besse, and B. Chaib-draa, "Solving the continuous time multiagent patrol problem," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2010, pp. 941–946.

[41] A. Marino, L. Parker, G. Antonelli, F. Caccavale, and S. Chiaverini, "A fault-tolerant modular control approach to multi-robot perimeter patrol," in *Proc. IEEE International Conf. on Robotics and Biomimetics (ROBIO)*, 2009, pp. 735–740.

[42] S. Darbha, K. Krishnamoorthy, M. Pachter, and P. Chandler, "State aggregation based linear programming approach to approximate dynamic programming," in *Proc. IEEE Conf. Decision and Control*, 2010, pp. 935–941.

[43] K. Krishnamoorthy, M. Pachter, P. Chandler, D. Casbeer, and S. Darbha, "UAV perimeter patrol operations optimization using efficient dynamic programming," in *Proc. American Control Conf.*, 2011, pp. 462–467.

[44] K. Krishnamoorthy, M. Pachter, P. Chandler, and S. Darbha, "Optimization of perimeter patrol operations using UAVs," *AIAA J. Guidance, Control and Dynamics*, vol. 35, no. 2, pp. 434–441, 2012.

[45] K. Krishnamoorthy, M. Pachter, S. Darbha, and P. Chandler, "Approximate dynamic programming with state aggregation applied to UAV perimeter patrol," *Internat. J. of Robust and Nonlinear Control*, vol. 21, no. 12, pp. 1396–1409, 2011.

[46] K. Krishnamoorthy, M. Park, S. Darbha, P. Chandler, D. Casbeer, and M. Pachter, "Lower bounding linear program for the perimeter patrol optimization problem," *AIAA Journal of Guidance, Control, and Dynamics*, in press, 2013.

[47] M. Park, S. Darbha, K. Krishnamoorthy, P. P. Khargonekar, M. Pachter, and P. Chandler, "Sub-optimal stationary policies for a class of stochastic optimization problems arising in robotic surveillance applications," in *Proceedings of the 5th Annual Dynamic Systems and Control Conference*, no. DSCC2012-8610. ASME, Oct 2012.

[48] A. Manne, "Linear programming and sequential decisions," *Management Science*, vol. 6, no. 3, pp. 259–267, 1960.

[49] F. d'Epenoux, "A probabilistic production and inventory problem," *Management Science*, vol. 10, no. 1, pp. 98–108, 1963.

[50] E. Porteus, "Bounds and transformations for discounted finite Markov decision chains," *Operations Research*, vol. 23, no. 4, pp. 761–784, 1975.

[51] K. Krishnamoorthy, M. Park, M. Pachter, P. Chandler, and S. Darbha, "Bounding procedure for stochastic dynamic programs with application to the perimeter patrol problem," in *Proc. American Control Conf.*, Montreal, QC, CA, June 2012, pp. 5874–5882.

[52] H. Takagi and L. Kleinrock, "A tutorial on the analysis of polling systems," University of California, Los Angeles, Department of Computer Science, Tech. Rep. CSD-85005, Feb 1985.